
INPUT
Publicación práctica para usuarios de MSX
Revista mensual 1986
Año 1 - Número 6 Precio 350 Ptas.
MSX
Puzzles
y matemáticas
Conoce todo sobre
READ y DATA
Un programa que aprende,
Inteligencia Artificial
Las tortugas del LOGO
entienden castellano
LISTA DE PREMIADOS
EN EL SORTEO
DE VERANO

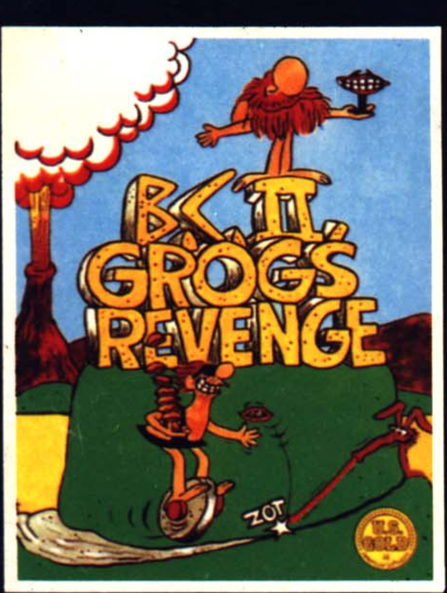
ERBE

NUESTROS JUEGOS ESTAN HECHOS PARA TI

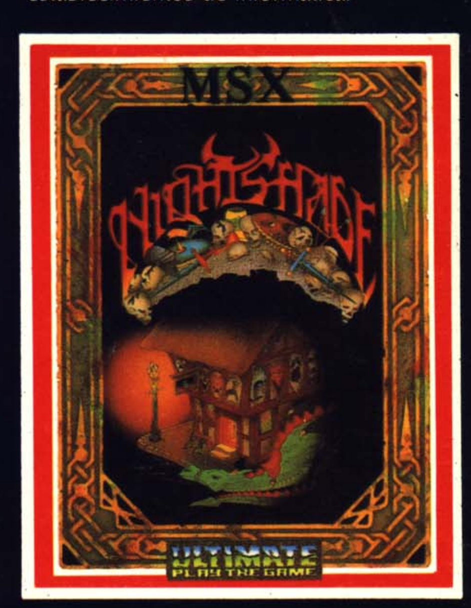
Como usuario de MSX, puede que hayas tenido la sensación de que las casas importantes de software te habían olvidado. Para remediar esa situación, ERBE, U. S. GOLD y ULTIMATE han conseguido convertir al MSX una serie de fabulosos programas pensados para aprovechar al máximo las grandes posibilidades de estos ordenadores. Aquí están cinco sobresalientes juegos en cassette que van a demostrarte lo que puede dar de sí tu MSX sin que tengas que gastar una fortuna en "cartuchos".
Directo desde Norte América llega GROG'S REVENGE. Personajes de comic inmensamente populares en U.S.A. Grog y Thor son dos trogloditas, que se desenvuelven como pueden en la Edad de Piedra... y es que la supervivencia no es fácil. Diplodocus, cavernas, rocas, simas y mil peligros les acechan en el juego más original, que jamás hayas visto. "Filmation", es una técnica de programación creada y desarrollada por Ultimate para producir juegos de una calidad única. NIGHTSHADE utiliza "Filmation II" y el resultado es un juego de proporciones, que superan todo lo hecho hasta el momento en gráficos tridimensionales. BOUNDER de Gremlin Graphics es el programa que va a poner a prueba tu habilidad en el manejo del joystick. Mover una pelota de tenis que rebota a través de un sinfín de pantallas en las que has de controlar los botes para evitar que caiga donde no debe, es una tarea de lo más difícil y adictivo y que ha hecho de este juego que sea N.º 1 en Inglaterra. Y por último DAMBUSTERS. Esta simulación no necesita presentaciones, se trata de una fiel reconstrucción de la legendaria misión, que llevó a cabo durante la II Guerra Mundial el Escuadrón 617, y en la que destruyeron la presa que daba energía a las fábricas de armamento alemanas. DAMBUSTERS con su mezcla de simulador, acción y estrategia es ya un clásico dentro de los programas de ordenador. Todos estos grandes juegos en cassettes están disponibles ahora para tu MSX en todos los grandes establecimientos de informática.

THE DAM BUSTERS









DISTRIBUIDOR EXCLUSIVO PARA ESPAÑA: ERBE SOFTWARE. C/. STA. ENGRACIA, 17
28010 MADRID. TEL. (91) 447 34 10 - DELEGACION BARCELONA, AVDA. MISTRAL, N.º 10 - TEL. (93) 432 07 31

AÑO 1 NUMERO 6



DIRECTOR:
Alejandro Diges
DIRECTOR TECNICO:
Roberto Menéndez
COORDINADOR EDITORIAL:
Francisco de Molina
DISEÑO GRAFICO:
Tomás López
COLABORADORES:
Antonio Taratiel, Luis R. Palencia, Francisco Tórtola, J. A. Feberero, Esther de la Cal, Ernesto del Valle, Equipo Molisoft, Javier Portillo.

INPUT MSX es una publicación juvenil de EDICIONES FORUM
GERENTE DIVISION DE REVISTAS:
Angel Sabat

PUBLICIDAD: José Real-Grupo Jota
Madrid: c/ Gral. Varela, 35, 3.º-11
Teléf. 270 47 02/03
Barcelona: Avda. de Sarriá, 11-13, 1.º
Teléf. 250 23 99

FOTOMECANICA: Ochoa, S. A.

COMPOSICION: EFCA, S. A.

IMPRESION: Sirven Grafic
C/ Gran Vía, 754-756. 08013 Barcelona
Depósito legal: B-21953-1986

SUSCRIPCIONES: EDISA,
López de Hoyos, 141. 28002 Madrid
Teléf. (91) 415 97 12

REDACCION:
Paseo de la Castellana, 93, 14.ª
28046 Madrid. Teléf. 456 54 13

DISTRIBUIDORA
R.B.A. PROMOTORA DE EDICIONES, S. A.
Travesera de Gracia, 56. Edificio Odiseus.
08006 Barcelona.

El precio será el mismo para Canarias que para la Península y en él irá incluida la sobretasa aérea.

INPUT MSX es una publicación controlada por



INPUT MSX

SUMARIO

NUEVAS TENDENCIAS

Abrumadora ha sido realmente la respuesta al sorteo que propusimos en el número especial de verano. Ahora llega el momento de comprobar si vuestro nombre aparece en la relación de afortunados. Si estáis incluidos, entonces enhorabuena. De lo contrario no desesperéis, es intención de quienes hacemos **INPUT** continuar ofreciendo la posibilidad de obtener algún premio lo más a menudo que permitan las circunstancias. Este es el caso del ganador en el concurso convocado por **Anaya Multimedia** desde la revista. Como resultado acudió a la feria **PCW** en Londres uno de los lectores más habilidosos descifrando claves. Según cuentan las crónicas se divirtió de lo lindo. Septiembre una vez más ha sido pródigo en la aparición de rumores y nuevos productos. La feria británica antes mencionada junto con el **Sonimag** han servido para apreciar cómo parecen confirmarse dos tendencias de cara al futuro de la microinformática. Por un lado los ordenadores destinados a juegos con más memoria, capacidades gráficas y sonido, siendo buena muestra el nuevo **Spectrum+2** y los **MSX 2.** Por otro lado, los ordenadores compatibles **PC** de bajo precio de venta, situándose al alcance de muchos más usuarios, sistemas que hasta hace poco eran un lujo asiático para quienes no fueran empresas o profesionales. El denominador común de ambas tendencias es la bajada de precios aplicada en septiembre. De lo que no cabe duda es que nos encontramos ante el umbral de una nueva etapa.

LOS MEJORES DE INPUT

Hemos pensado que es interesante disponer de un ***ranking*** que ponga en claro, mes a mes, cuáles son los programas preferidos de nuestros lectores. Para ello, es obligado preguntaros directamente y tener así el mejor termómetro para conocer vuestras preferencias. Podéis votar por cualquier programa aunque no haya sido comentado todavía en **INPUT**.

El resultado de las votaciones será publicado en cada número de **INPUT**.

Entre los votantes sortearemos 10 cintas de los títulos que pidáis en vuestros cupones.

Nota: No es preciso que cortéis la revista, una copia hecha a máquina o una simple fotocopia sirven.

Enviad vuestros votos a: **LOS MEJORES DE INPUT** P.º de la Castellana, 93. Planta 14. 28046 Madrid

ELIGE TUS PROGRAMAS

Primer título elegido ____________________

Segundo título elegido ____________________

Tercer título elegido ____________________

Programa que te gustaría conseguir ____________________

Qué ordenador tienes ____________________

Nombre ____________________

1.er Apellido ____________________

2.º Apellido ____________________

Fecha de nacimiento ____________________

Teléfono ____________________

Dirección ____________________

Localidad ____________________

Provincia ____________________

INPUT MSX N.º 6

PHILIPS A POR TODAS

El modelo MSX2 VG-8235 de Philips ha dejado de ser el único ejemplar de nueva generación de la firma holandesa. De golpe la oferta se amplia con la gama recientemente presentada en Sonimag, de nombre NMS (New Media Systems).

Los tres nuevos equipos presentados bajo estas siglas son el NMS 8220, el 8250 y el 8280 (Video Computer). Es este último el que más expectación despertó, y es lógico si tenemos en cuenta que se trata de algo totalmente nuevo.

En efecto, mientras que los modelos 8220 y 8250 son versiones más o menos renovadas de equipos MSX2, el *Video Computer* es, más que un ordenador, un sistema completo de adquisición, almacenamiento y tratamiento de imágenes de vídeo. El equipo permite la edición de las cintas de vídeo, incluyendo en ellas textos, mezclando y superponiendo distintas imágenes, creando e incorporando distintos efectos especiales, etc. Para estos increíbles cometidos, el sistema incorpora 128K de RAM de usuario y otros 128K de RAM de vídeo, así como una unidad de *diskettes* de 1M. Por supuesto, además de convertir el salón de casa en un estudio de vídeo, el sitema permite utilizar todo el *software* MSX, al ser compatible con el resto de las máquinas del estandar.

KONAMI JUEGA FUERTE

Konami, que hace poco anunció su intención de fabricar y distribuir sus propios programas para Europa, tras su aventura con Imagine, no ha dejado pasar el tiempo.

Para los próximos meses, el distribuidor de los japoneses para nuestro país, Serma, aununcia los siguientes lanzamientos:

El superhit Green Beret en versión MSX, Jail Break, que narra la fuga de un grupo de presidiarios, Basketball para recordar los mundiales de baloncesto, Nightmare; una aventura gráfica, la version MSX de The Goonies, Iron Horse; uno de los arcades de más éxito en el 86, Yie Ar Kung Fu II, Sao Lin Road y Nemesis.

Todos los títulos aparecerán en cartucho para MSX.

También se está preparando la versión de Salamander, un programa que se anuncia como el Mega Juego del 1987.

Un buen montón de novedades que iremos comentando en próximos números.

LOCOS POR LA MUSICA

Con el nuevo sistema Philips, de nombre Music Module, se amplian de forma espectacular los horizontes musicales de los usuarios de MSX.

El sistema incluye un cartucho con un sintetizador FM de nueve canales de sonido independientes, con 60 sonidos pregrabados y un conjunto de 20 ritmos de acompañamiento (que incluyen estilo pop, clásico, español, etc)

El conjunto incorpora además un muestreador, con el que sera posible grabar, manipular y reproducir posteriormente cualquier sonido externo, que llegue al sistema a través del micrófono incorporado o de la entrada de audio. A una frecuencia de muestreo de 16KHz, el muestreador permite almacenar hasta 4 segundos de grabación, transferibles a *cassette* o *diskette*.

El módulo permite asimismo el control de tono o la incorporación de un eco, de retardo variable, con los que hacer experimentos en el campo de los efectos especiales.

Por si fuera poco, se ha incluido un interface MIDI (el estandar musical) con conexiones *In*, *Out* y *Thru*.

Todas estas opciones se manejan desde 8 menús básicos, presentados mediante una serie de iconos.

MAS SOFT PARA MSX2

Un buen montón de programas, especialmente diseñados para aprovechar las posibilidades gráficas de la segunda generación MSX, es lo que nos ofrece Idealogic. Para empezar nos encontramos con la colección Telarium; un conjunto de aventuras gráficas interactivas, con argumentos basados en novelas de éxito, que al parecer ya estan disponibles en nuestro mercado. Los títulos anunciados son: Cita con Rama, de Arthur Clarcke, El Mago de Oz, de L. Frank, Dragon World, La Isla del Tesoro, Farenheit 451, Amazon y por ultimo Perry Mason en El caso del asesinato en el Mandarin.

Aparte de esta serie, Idealogic presenta un atractivo paquete, de nombre Aerobic, para MSX2, destinado a los que quieren mantenerse en forma. El programa utiliza profusamente los gráficos, y una serie de técnicas de animación, para presentar y explicar los distintos ejercicios. Una serie de temas musicales, compuestos expresamente para este *soft* por un buen profesional de la música, acompañan y marcan el ritmo de los distintos ejercicios.

Ludicbit es el nombre de otro de los lanzamientos anunciados. Se trata de una colección de títulos, fundamentalmente de arcades, para MSX1 y 2, entre los que cabe destacar una versión pinball 3D, catalogada como la primera del mercado mundial.

La lista de novedades termina con dos paquetes de utilidad; Supersprites y Side Pack, el primero destinado a la edición y animación de *sprites* y el segundo a servir de calculadora, listín telefónico y block de notas.



LA MEMORIA DE VIDEO DE MSX: SCREEN 3

El modo de gráficos de baja resolución (SCREEN 3) tiene, al menos aparentemente, poca utilidad. Los gráficos en este modo de pantalla son bastante grotescos, sin embargo, resulta interesante para la programación de juegos y, sobre todo a la hora de hacer grandes y vistosos rótulos.

Además de la parte de memoria destinada a contener la información relativa a los *sprites*, sólo existen en este modo de pantalla dos bloques de memoria para almacenar toda la información necesaria.

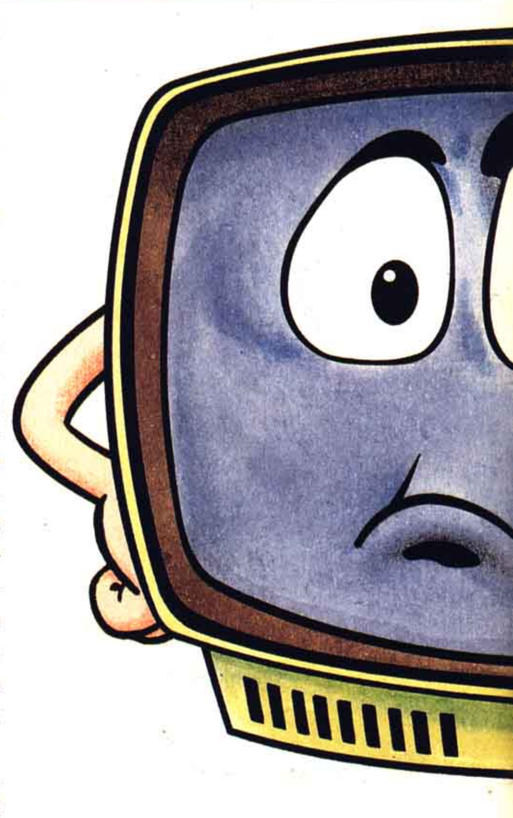
PUNTOS EN SCREEN 3

En este modo de pantalla hay un bloque de memoria VRAM que contiene la información relativa a la representación de puntos en la pantalla.

Este trozo de memoria se llama **Tabla del generador de patrones** y ocupa 1536 bytes. Su dirección de comienzo viene dada por BASE (17) que es igual a 0.

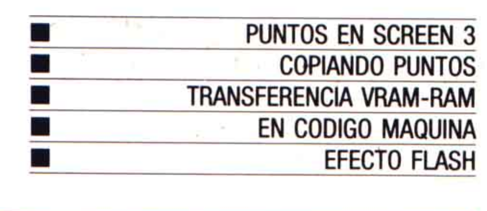
En este modo 3, los puntos de la imagen o *pixels* tienen una superficie que es 16 veces la de los *pixels* del modo 2 (gráficos de alta resolución). Es decir, el lado de un *pixel* es 4 veces el lado de un *pixel* en SCREEN 2: la x (eje horizontal de izquierda a derecha) varía de 0 a 255 y la y (eje vertical de arriba a abajo) varía de 0 a 191. Lo que ocurre es que en lugar de tener 256 *pixels* horizontales por 192 verticales (en total 49152 puntos), tenemos 64 *superpixels* horizontales por 48 verticales (en total 3072 superpuntos), y si situamos un punto en las coordenadas (100,101), el ordenador dibuja un *superpixel* en el punto correspondiente al par de coordenadas que se obtienen como el entero resultado de dividir la x y la y del punto en cuestión por 4, es decir, (100/4,101/4) = (25,25).

Para estudiar la tabla 17 imaginemos la pantalla como una trama de 32



```
                         ;RUTINA PARA VOLCAR TABLAS
                         ;17 Y 15 DE VRAM A RAM
                                 ORG  0A368H
                                 LOAD 0A368H
                         ;TABLA 17
A368 210000                      LD   HL,0            ;INICIO VRAM
A36B 010006                      LD   BC,1536         ;LONGITUD
A36E 1151C3                      LD   DE,50001        ;INICIO RAM
A371 CD5900                      CALL 0059H           ;MUEVE BLOQUE

                         ;
                         ;TABLA 15
A374 210008                      LD   HL,2048         ;INICIO VRAM
A377 010003                      LD   BC,768          ;LONGITUD
A37A 1152C9                      LD   DE,51538        ;INICIO RAM
A37D CD5900                      CALL 0059H           ;MUEVE BLOQUE
A380 C9                          RET
                                 END
```



- PUNTOS EN SCREEN 3
- COPIANDO PUNTOS
- TRANSFERENCIA VRAM-RAM
- EN CODIGO MAQUINA
- EFECTO FLASH

cuadros horizontales que numeraremos de 0 a 31 y 24 cuadros verticales que numeraremos de 0 a 23, tal como hacíamos al hablar de SCREEN 2 (ver figura 1). Supongamos ahora la pantalla dividida en 6 franjas horizontales de 32 cuadros horizontales por 4 verticales.

La figura 2 representa la franja superior (franja número 1). En ella vemos que cada cuadro está dividido en 2 mitades, una encima de la otra y que numeramos con 0 (la superior) y 1 (la inferior) . A cada una de estas 2 mitades le corresponde un byte y 2 *superpixels*. Cada *superpixel* se corresponde con una mitad del byte; el *superpixel* de la izquierda con los 4 bits de mayor peso (al que asignaremos el número 1) y el de la derecha con los 4 de menor peso (número 0). Así en cada byte se almacenan dos colores, correspondientes a cada uno de los *superpixels*. Esta es toda la información que contienen los bytes de esta tabla.

Al dar la orden SCREEN 3, todos los *nibbles* (un *nibble* es medio byte) de todos los bytes de esta tabla toman el número (entre 0 y 15) correspondiente al color del fondo (al conectar el ordenador, el azul=4).

Veamos ahora como se corresponden cada pareja de *superpixels* de la pantalla con los bytes de esta tabla: Al byte 0 le corresponde la mitad 0 (superior) del cuadro (0,0), al byte 1 la mitad 1 (inferior) del mismo cuadro, al byte 2 la mitad 0 del cuadro (0,1), que es el situado justo debajo del (0,0), y así hasta el byte 7 que está emparejado con la mitad 1 del cuadro (0,3). Los colores de la mitad 0 del cuadro (1,0) están almacenados en el byte 8... y así hasta la mitad 1 del cuadro (31,3) que se almacena en el byte 255. El byte 256 almacena los colores de la mitad 0 del cuadro (0,4),... y el byte 1535 los de la mitad 1 del cuadro (31,23).

Para saber que byte le corresponde al *superpixel* l de la mitad k del cuadro (i,j) basta aplicar la fórmula:

```
(j/4)*256+i*8+2*(jMOD4)+K
```

Si l=0 se tratará del *nibble* bajo y si l=1 se tratará del *nibble* alto.

Si por ejemplo queremos dibujar un

```
                        ;RUTINA PARA VOLCAR TABLAS
                        ;17 Y 15 DE RAM A VRAM
                                ORG  0A368H
                                LOAD 0A368H
                        ;TABLA 17
A368 2151C3                     LD   HL,50001           ;INICIO RAM
A36B 010006                     LD   BC,1536            ;LONGITUD
A36E 110000                     LD   DE,0               ;INICIO VRAM
A371 CD5C00                     CALL 005CH              ;MUEVE BLOQUE
                        ;
                        ;TABLA 15
A374 2152C9                     LD   HL,51538           ;INICIO RAM
A377 010003                     LD   BC,768             ;LONGITUD
A37A 110008                     LD   DE,2048            ;INICIO VRAM
A37D CD5C00                     CALL 005CH              ;MUEVE BLOQUE
A380 C9                         RET
                                END
```

punto de color negro en el *superpixel* 1 de la mitad 1 del cuadro (20,19) basta escribir el valor hexadecimal 14 (1=color negro, 4=azul color del fondo) en el byte:

```
(19/4)*256+20*8+2*(19MOD4)
+1=1191
```

lo que podemos hacer con auxilio de la instrucción VPOKE, así:

```
10 SCREEN 3
20 VPOKE 1191,&H14
30 GOTO 30
```

CUADROS IGUALES

Al hablar de SCREEN 2 veíamos que podíamos dividir la pantalla gráfica en 3 franjas horizontales en las que podíamos repetir cuadros.

En el modo multicolor que estamos tratando, disponemos de la **Tabla de Nombres de Patrones** (cuya dirección de comienzo viene dada por BASE (15)=2048). Esta tabla está dividida en 4 zonas, pero estas zonas no parecen estar muy ordenadas. Cada zona se corresponde con 6 líneas horizontales de 32 cuadros cada una (de la cuadrícula de la que hablamos al principio), de forma que la primera zona (zona 1) está formada por los 32 cuadros de las filas 0, 4, 8, 12, 16 y 20, la zona 2 por las filas 1, 5, 9. 13, 17 y 21, la zona 3 por las filas 2, 6, 10, 14, 18 y 22 y la zona 4 por las filas 3, 7, 11, 15, 19 y 23. Así cada zona ocupa en memoria (a razón de un byte por cuadro) un total de 32*6=192 bytes, y la tabla 15 completa ocupa un total de 192*4=768 bytes. Al dar la orden SCREEN 3, los bytes correspondientes a la primera fila de cada zona tienen almacenados los números 0 a 31, los correspondientes a la segunda fila de cada zona, los números 32 a 63... y los correspondientes a la fila sexta de cada zona, los números 160 a 191. De esta forma los números correspondientes a cada cuadro de la cuadrícula son los que se indican en la figura 3.

Podemos conocer el byte que corresponde al cuadro (i,j) con auxilio de la relación:

```
2048+j*32+i
```

y el valor que, en principio estará almacenado en ese byte será:

```
32*(j/4)+i
```

Modificando los valores de esta tabla podemos repetir puntos en la pantalla, pero siempre dentro de cada zona. Es decir, al igual que ocurría en SCREEN 2, no se pueden repetir puntos de zonas distintas.

Como ejemplo, veremos cómo utilizando esta tabla podemos escribir un rótulo repetido 6 veces en la pantalla. Lo que hacemos es escribir en los bytes correspondientes a las líneas 4 a 23 los valores 0 a 31, que son los almacenados en los bytes correspondientes a las líneas 0 a 3.

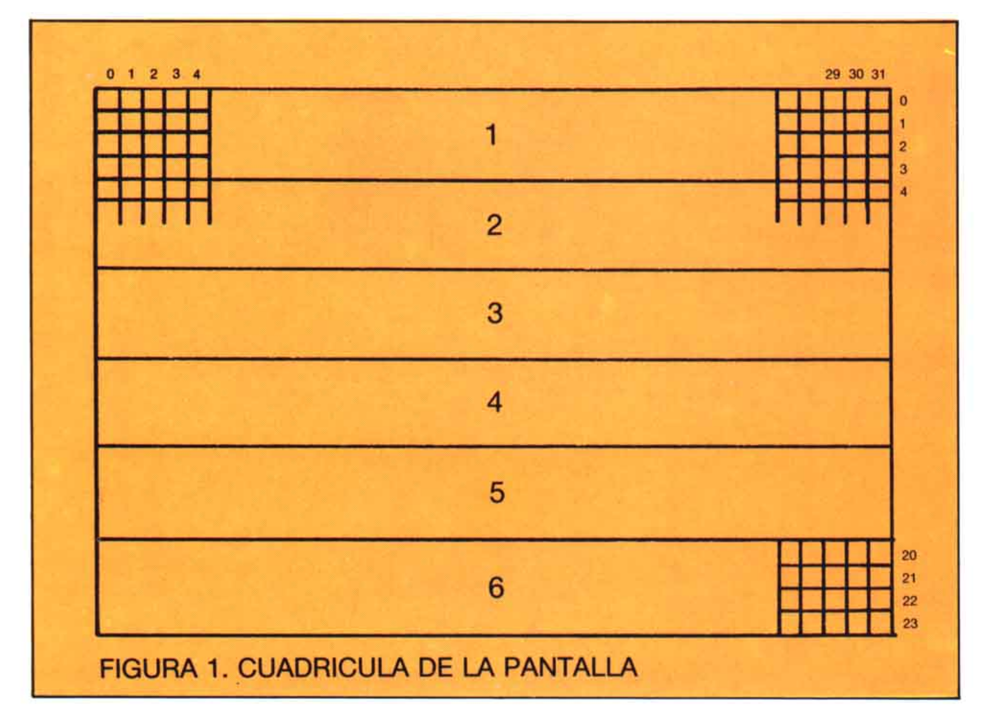


FIGURA 1. CUADRICULA DE LA PANTALLA

APLICACIONES EN CODIGO MAQUINA

La primera aplicación va a consistir en algo que ya hicimos al hablar del modo 2, es decir en pasar la VRAM a RAM. Hagámoslo en BASIC. Como consideración a aquellos que dispongan de unidad de disco, almacenaremos los datos a partir de la posición 50000.

La subrutina que pasa las tablas 17 y 15 a RAM, que podemos incluir en nuestros programas es:

```
10 CLEAR 200,50000!
50000 FOR I=0 TO 1535
50010 POKE 50001!+I,
      VPEEK(I)
50020 NEXT I
50030 FOR I=2048 TO 2815
50040 POKE 49490!+I,
      VPEEK(I)
50050 NEXT I
50060 RETURN
```

y la que pasa la RAM a VRAM es:

```
10 CLEAR 200,50000!
60000 FOR I=0 TO 1535
60010 VPOKE I,PEEK
      (50001!+I)
60020 NEXT I
60030 FOR I=2048 TO
      2815
60040 VPOKE I,PEEK
      (49490!+I)
60050 NEXT I
60060 RETURN
```

Veamos como trabajan estas subrutinas. Incluyamos ambas subrutinas en el siguiente programa:

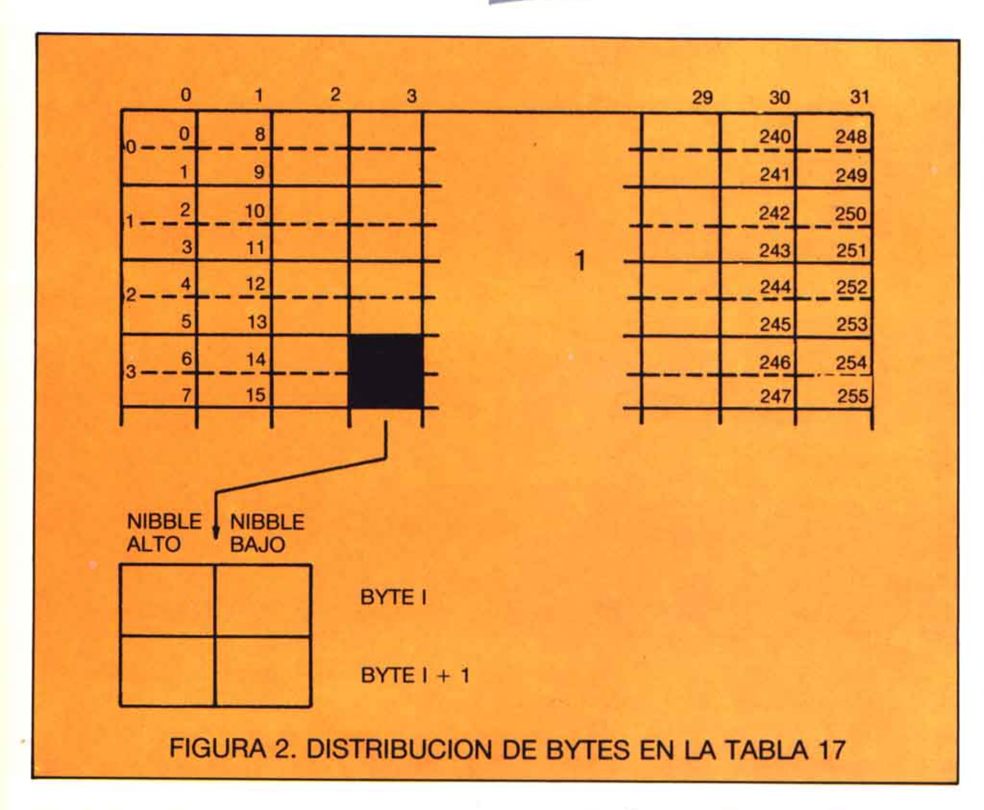


FIGURA 2. DISTRIBUCION DE BYTES EN LA TABLA 17

```
10 CLEAR 200,50000!
20 SCREEN 3
30 OPEN "GRP:" AS #1
40 PRESET (64,0)
50 PRINT#1,"HOLA"
60 FOR I=2176 TO
   2815 STEP 32
70 FOR J=0 TO 31
80 VPOKE I+J,J
90 NEXT J
100 NEXT I
110 GOSUB 50000
120 SCREEN 3
130 GOSUB 60000
140 IF INKEY$=""
    THEN 140
150 STOP
```

Con las sentencias dadas entre las líneas 20 y 100, se escribe en la pantalla la palabra «HOLA» en gran tamaño seis veces (hemos utilizado la tabla 15 para repetir patrones). En la línea 110 se almacena la VRAM en RAM (subrutina 50000 a 50060). En la 120 se actualiza la pantalla y en la línea 130 la RAM pasa de nuevo a la VRAM (subrutina 6000 a 60060).

El siguiente programa es similar al anterior, pero realiza las operaciones de guardar la VRAM en RAM y la inversa mediante sendas rutinas en código máquina dadas por los Datas de las líneas 10000 y 10010. Además incluimos en estas páginas el código ensamblador de ambas rutinas.

```
10 CLEAR 200,50000!
20 FOR I=1 TO 50
30 READ A$
40 POKE 52310!+I,
   VAL("&H"+A$)
50 NEXT I
60 DEFUSR0=52311!
70 DEFUSR1=52336!
80 SCREEN 3
90 OPEN "GRP:" AS#1
100 PRESET (64,0)
110 PRINT#1,"HOLA"
120 FOR I=2176 TO
    2815 STEP 32
130 FOR J=0 TO 31
140 VPOKE I+J,J
150 NEXT J
160 NEXT I
170 U=USR(0)
180 FOR I=1 TO 100
190 BEEP
200 NEXT I
210 SCREEN 3
220 U=USR1(0)
230 BEEP
240 IF INKEY$="" THEN 240
250 STOP
10000 DATA 21,00,00,01,00,06,
      11,51,C3,CD,59,00,21,00
      ,08,01,00,03,11,52,C9,C
      D,59,00,C9
10010 DATA 21,51,C3,01,00,06,
      11,00,00,CD,5C,00,21,52
      ,C9,01,00,03,11,00,08,C
      D,5C,00,C9
```

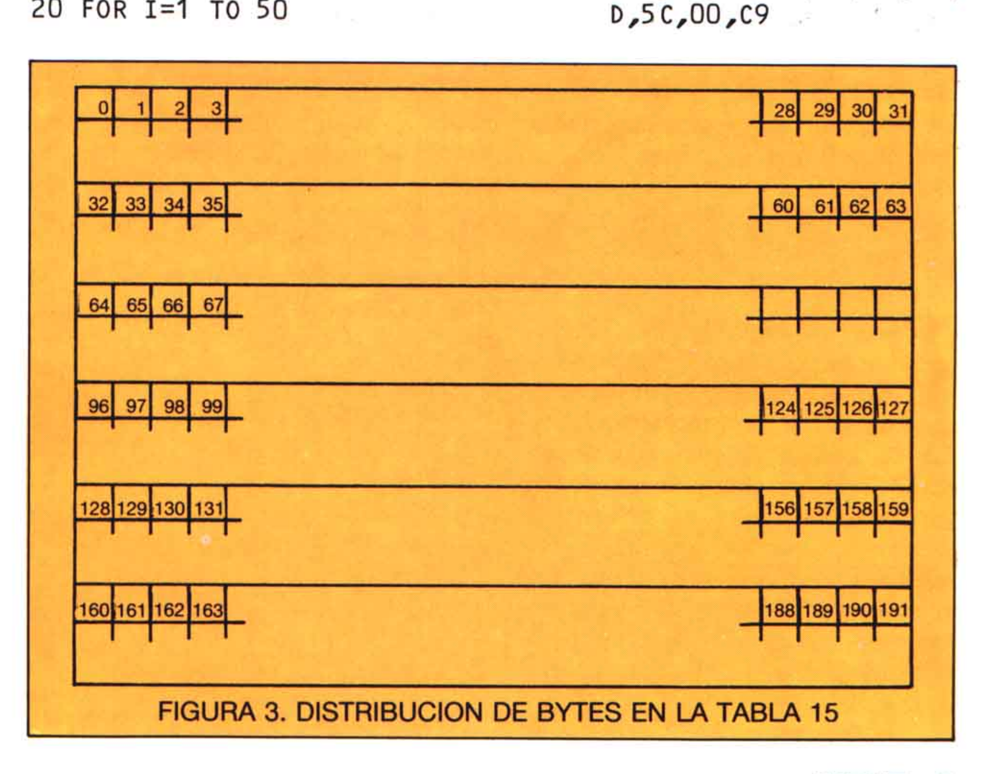


FIGURA 3. DISTRIBUCION DE BYTES EN LA TABLA 15

```
                    ;RUTINA PARA INVERTIR COLOR
                    ;EN SCREEN3
                    ;REPITE CAMBIO 100 VECES
                    ;
                                ORG  0A4F1H
                                LOAD 0A4F1H
                    ;
A4F1 DB99                       IN   A,(99H)
A4F3 016400                     LD   BC,100
A4F6 0B             PRINC:      DEC  BC             ;COMIENZA CUENTA
A4F7 21FF00                     LD   HL,0FFH        ;RUTINA RETARDO
A4FA 2B             RETARDO:    DEC  HL
A4FB 7C                         LD   A,H
A4FC B5                         OR   L
A4FD 20FB                       JR   NZ,RETARDO     ;FIN RETARDO
A4FF 210006                     LD   HL,1536        ;CAMBIO COLOR
A502 2B             CAMBIA:     DEC  HL
A503 CD4A00                     CALL 4AH            ;LEER VRAM
A506 2F                         CPL
A507 CD4D00                     CALL 4DH            ;ESC. EN VRAM
A50A 7C                         LD   A,H
A50B B5                         OR   L
A50C 20F4                       JR   NZ,CAMBIA
A50E 78                         LD   A,B            ;COMPRUEBA SI 100
A50F B1                         OR   C
A510 20E4                       JR   NZ,PRINC
A512 C9                         RET
                                END
```

Las rutinas en BASIC tardan, cada una 17.24 seg. en ejecutarse, mientras que las rutinas en código máquina tardan ¡¡¡0.04!!! seg. cada una.

EFECTO FLASH

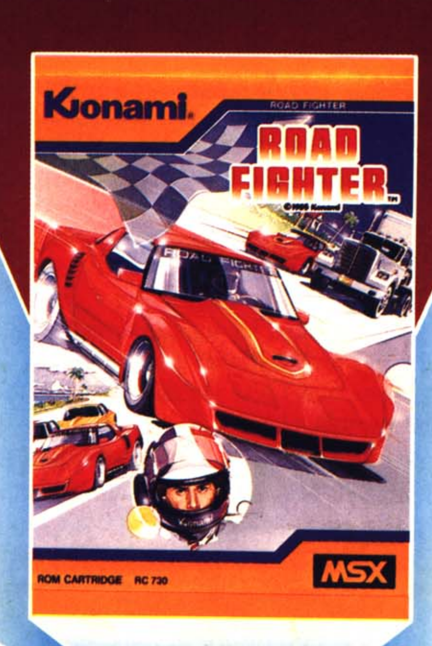
La segunda aplicación que vamos a ver sirve para conseguir un efecto de centelleo, por ejemplo en el rótulo de presentación de un programa. La rutina en código máquina correspondiente cambia, alternativamente los colores del fondo y de la tinta en SCREEN 3, alterando el contenido de la tabla 17. Lo que hace la rutina es poner en todos los bytes de esta tabla el número complemento a dos del que había inicialmente. El siguiente programa BASIC incluye un DATA en la línea 10000 con los códigos hexadecimales de dicha rutina. En este programa hemos inicializado los colores del fondo y de la tinta al negro y blanco, respectivamente, colores con los que resalta más el efecto. La operación de cambio se realiza 100 veces antes de devolver el control al BASIC. También incluimos el código ensamblador para esta rutina.

```
10 CLEAR 200,50000!
20 FOR I=1 TO 34
30 READ A$
40 POKE 50000!+I,
   VAL("&H"+A$)
50 NEXT I
60 DEFUSR=50001!
70 COLOR 15,1,1
80 SCREEN 3
90 OPEN "GRP:" AS#1
100 PRESET (64,0)
110 PRINT #1,"HOLA"
120 FOR I=2176 TO
    2815 STEP 32
130 FOR J=0 TO 31
140 VPOKE I+J,J
150 NEXT J
160 NEXT I
170 U=USR(0)
180 STOP
10000 DATA DB,99,01,64,00,0B,
      21,FF,00,2B,7C,B5,20,FB
      ,21,00,06,2B,CD,4A,00,2
      F,CD,4D,00,7C,B5,20,F4,
      78,B1,20,E4,C9
```

Si quieres modificar la velocidad a la que se produce el centelleo, no tienes más que modificar los valores 4, 6, 7 y 8 de la sentecia DATA (64, 00, FF y 00 respectivamente). Prueba diversos valores y observa los resultados.

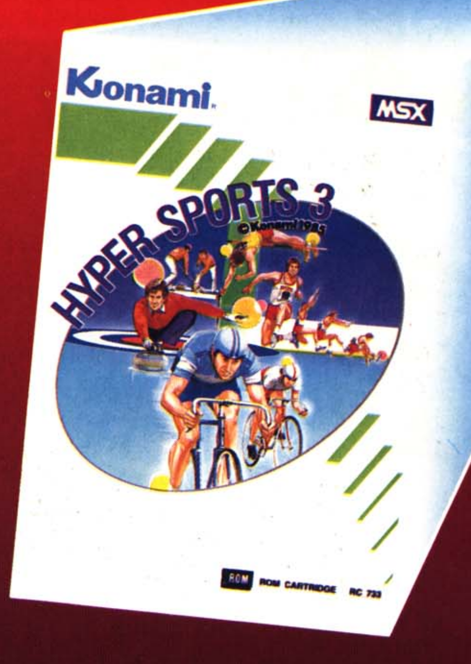
LO MEJOR EN MSX







Konami







VISITE LA DIVISION 



GALERIAS

TODO SOBRE READ Y DATA

- COMO FUNCIONAN READ Y DATA
- LA SENTENCIA RESTORE
- COMO CONSTRUIR GRAFICOS SENCILLOS A PARTIR DE LOS DATA

Haciendo que el ordenador lea por medio de la sentencia READ un gran volumen de datos te puedes ahorrar tener que teclear programas largos. Puedes usar esta misma técnica para todo, desde gráficos hasta índices sencillos.

Una de las características que hace que un ordenador sea tan versátil es el uso de variables. Y casi siempre la asignación de valores a las variables es algo extremadamente sencillo, no tienes más que escribir, por ejemplo, X = 5. Pero hay ocasiones en que la cantidad de información que se quiere utilizar es tan grande que llega a desbordarte. Aquí es donde encuentran su gran utilidad las sentencias DATA y sus acompañantes READ y RESTORE.

La palabra DATA tiene aquí un significado muy específico. Se suele utilizar el término «datos» para designar una gran variedad de cosas. Por ejemplo, un programa, una rutina en código máquina y una matriz almacenada en cinta o en *diskette*, a todo esto se le suele llamar a veces «datos». Pero en todo este artículo al hablar de datos, nos estamos refiriendo a las sentencias DATA y a los elementos que contienen.

El primer método que se suele aprender para asignar valores a una variable, ya se trate de una variable numérica o de una cadena de caracteres, suele ser por medio de una sentencia INPUT. Pero ésta sólo resulta de utilidad cuando es el propio usuario el que suministra la información al ordenador cada vez que se ejecuta un programa.

Muchas veces sin embargo, hay valores fijos que no tienen que ser introducidos ni modificados por el operador, por lo que se pueden incorporar permanentemente en el programa.

En este caso y mientras el volumen de datos a introducir no sea demasiado grande, se recurre a las sentencias de asignación del tipo:

```
10 X=5
11 LET X=5
12 A$="ABC"
13 LET A$="ABC"
```

En ellas, todo lo que hay a la derecha del signo = queda asignado a la variable que figura a la izquierda de dicho signo. Como ves, puedes omitir la palara clave LET sin que haya ningún problema. Es más, si no la utilizas ahorras memoria. Se ha incluido en el BASIC **MSX** simplemente porque es una palabra clave de uso común en otras versiones de este lenguaje, y lo que **Microsoft** pretendía era crear una versión BASIC lo más completa y estandarizada posible.

El problema aparece cuando se trabaja con un gran volumen de datos. Utilizar grandes masas de sentencias de asignación, además de tedioso a la hora de programar, resultaría bastante ineficiente, haría más lenta la ejecución del programa y consumiría un innecesario espacio de memoria.

Aquí tienes un ejemplo en el que se lleva a cabo la asignación de los valores de cuatro variables, que un supuesto programa utiliza a la hora de la presentación de documentos:

```
10 A$="FECHA"
20 B$="CLIENTE"
30 C$="NO.CLIENTE"
40 D$="EMPRESA"
50 PRINT A$,B$,C$,D$
```

Para hacer esto mismo podrías utilizar sentencias DATA. El programa quedaría así:



```
10 READ A$,B$,C$,D$
20 PRINT A$,B$,C$,D$
30 DATA FECHA,CLIENTE,
   NO.CLIENTE,EMPRESA
```

A la vista del ejemplo, piensa por un momento en lo que ocurriría si, en lugar de cuatro encabezamientos, el programa tuviera que utilizar cincuenta o cien, todos ellos diferentes. Con el primer programa sería necesario incluir un montón de nuevas sentencias de asignación, mientras que con la segunda versión sólo harían falta una o dos más, dependiendo del ordenador y del tamaño de los encabezamientos.

COMO FUNCIONAN LAS SENTENCIAS «READ» Y «DATA»

¿Cómo funcionan realmente las sentencias READ y DATA? Cuando el ordenador se encuentra con una instrucción READ, explora todo el programa hasta que encuentra la primera sentencia DATA. A continuación asigna el valor del primer dato de dicha sentencia a la variable que figura en la sentencia READ.



Así, en el programa anterior la cadena de caracteres «FECHA» queda asignada a la variable A$, a continuación se asigna «CLIENTE» a la variable B$, etc. Cuando se han leído todos los valores de una sentencia DATA el ordenador pasa a la sentencia DATA siguiente.

Las sentencias DATA aparecen normalmente al final de los programas para «no estorbar» mientras el resto del programa se encuentra en fase de escritura y depuración. Pero realmente pueden ir situadas en cualquier parte. El ordenador simplemente ignora todas las sentencias DATA a menos que una sentencia READ le ordene hacer lo contrario. El siguiente programa funcionaría perfectamente:

```
10 DATA ALEMANIA
20 READ A$,B$
30 DATA FRANCIA,ITALIA,
   BELGICA
40 READ C$,D$
50 PRINT A$,B$,C$,D$
```

Sin embargo es evidente que el programa resultará mucho más fácil de leer si todas las sentencias DATA se ponen agrupadas en alguna parte.

Un programa puede contener tantas sentencias DATA como sea necesario. Lo único a tener en cuenta es un par de reglas sencillas que hay que respetar: Los datos de las sentencias DATA deben ir separados por comas y colocados en el orden en que el programa tenga que leerlos. Se pueden incluir tantos datos como se quiera en cada sentencia DATA, pero teniendo en cuenta que el número de caracteres en una línea no puede exceder nunca de 255. En cuanto la línea esté llena se empieza otra nueva; el ordenador sigue tomándolas en orden.

DIFERENTES TIPOS DE «DATA»

Hasta ahora hemos estado hablando de sentencias DATA que contenían cadenas de caracteres, pero también pueden contener números. Lo que no puedes hacer es incluir variables o expresiones aritméticas en tus sentencias DATA. Aunque ello resulte posible en las versiones BASIC de otros ordenadores, los **MSX** no contemplan este caso. Por ello si escribes algo como esto:

```
10 DATA A*5
```

ó

```
10 DATA 7+3
```

e intentas asignar estos valores a variables númericas obtendrás un bonito «*Syntax error*».

Las variables que aparezcan en la sentencia READ tienen que aparecer en el mismo orden y ser del mismo tipo que los elementos de la sentencia DATA. Por ejemplo, para una sentencia DATA como la siguiente:

```
10 DATA PARIS,1235,ROMA
```

es necesario una sentencia READ de este tipo:

```
10 READ A$,X,B$
```

Como puedes observar, la primera variable es alfanumérica (termina en $) lo que se corresponde con el primer dato que figura en la sentencia DATA; la cadena de caracteres PARIS. Lo mismo ocurre con la variable B$ y con el dato ROMA. Sin embargo el segundo dato es un valor numérico (1235), por lo que la variable correspondiente de la sentencia READ es una variable numérica (x).

PROBLEMAS QUE PRESENTAN LAS LINEAS «DATA»

Es fácil cometer errores al introducir los datos de una sentencia DATA. Los principales problemas se presentan cuando no tienes suficientes elementos, o cuando intentas que la sentencia READ lea datos de tipo equivocado. Si la sentencia DATA del ejemplo anterior hubiera sido tecleada erróneamente de la forma DATA 1235, PARIS, ROMA te encontrarías en primer lugar con que la variable A$ leería el dato numérico 1235. En este caso no se detectaría ningún tipo de error, pero se estaría considerando erróneamente el dato numérico como una cadena de caracteres. Otro caso distinto se plantea cuando se lee el si-

guiente dato. En este caso la variable numérica X intenta leer un dato no numérico (PARIS). Aquí ya no hay confusión posible; el ordenador detecta el error y lo indica mediante un mensaje del tipo *«Syntax error in...»* indicando el número de la línea DATA en la que se ha detectado el problema. Como verás donde más cuidado hay que tener es en el primer caso, pues el error que se produce no lo detecta el ordenador.

Otro de los problemas más frecuentes al trabajar con sentencias DATA es suministrar menos datos de los necesarios. Esto ocurre muchas veces cuando estás utilizando un bucle que contiene la sentencia READ, como ocurre en el programa que veremos a continuación. En cuanto cometas un error en el parámetro del bucle, o te dejes fuera accidentalmente alguno de los datos de las sentencias DATA (por ejemplo, si en lugar de una coma incluyes un punto para separar dos valores, el ordenador creerá que se trata de un sólo valor) la máquina intentará leer valores más allá del final de la lista de datos. El resultado será la detención del programa y la aparición de un mensaje indicando que faltan datos *«Out of Data in* (número de línea)». Este tipo de error crea mucha confusión entre los principiantes pues el número de línea que aparece en el mensaje, corresponde a la línea de la sentencia READ y no a la de la sentencia DATA que es donde hay que llevar a cabo la correción.

UNA LISTA SENCILLA

Aquí tienes un programa que utiliza un bucle para leer los datos con las sentencias READ y DATA. Es un programa muy sencillo de lista de teléfonos. Tú introduces el nombre de la persona y el ordenador te presenta su número de teléfono.

```
5 CLS
10 LOCATE 6,2:PRINT
   "DIRECTORIO TELEFONICO"
20 INPUT"Escribe el nombre"
   ;R$
30 FOR J=1 TO 5
40 READ N$,T$
50 IF N$=R$ THEN LOCATE 1,10:
   PRINT"EL NUMERO DE ";N$;
   " ES ";T$:STOP
60 IF N$="FIN" THEN LOCATE
   3,10:PRINTR$;" NO ESTA EN
   LA LISTA"
70 NEXT J
500 DATA PEPE,452 12 34,LUIS,
    876 23 45,ARTURO,567 34 5
    6,RAUL,784 34 12,FIN,FIN
```

Teclea el programa, pero ten la precaución de utilizar en la línea DATA los nombres y números de teléfono de tus amigos. Puedes poner todos los datos que quieras. Lo único que tienes que cuidar al final es de ajustar correctamente el contador de bucle de la línea 30.

Observa que la lista de datos se termina con FIN,FIN. De esta forma la línea 60 puede comprobar si se ha alcanzado el final de la línea. Si el programa ha podido leer hasta aquí, significa que no ha encontrado el nombre, por lo que imprime un mensaje para decírtelo. Hay que introducir «FIN» dos veces, debido a que la línea 40 lee los datos de dos en dos, con lo que de no ser así obtendrás un mensaje de error de falta de datos.

Fíjate también que para leer un número de teléfono se considera a éste como una cadena de caracteres. Se ha hecho así para respetar un signo de puntuación, el guión, que evidentemente no puede figurar como un número. Las cadenas de caracteres que figuren dentro de una sentencia DATA pueden contener espacios, pero no pueden contener comas. Como los diferentes datos están separados por comas, el ordenador consideraría la coma como el final de uno de los datos. Si te vieras obligado a introducir en una sentencia DATA una cadena de caracteres que contenga una coma, tienes que encerrarla entre comillas. Sería necesario hacer esto en una sentencia DATA como la que figura en el ejemplo siguiente:

```
10 READ N$,A$,B$,C$
20 DATA LUIS PELAEZ,
```

```
   "C/PINTO, 25",MURCIA,2345
30 PRINT N$,A$,B$,C$
```

USO DE «RESTORE»

Con los métodos que llevamos explicados hasta este momento, un programa es capaz de leer mediante una sentencia READ la correspondiente lista de datos de la sentencia DATA, pero sólo lo hará una vez a menos que se vuelva a ejecutar de nuevo el programa. Si quieres que se vuelvan a leer los datos, debes incorporar en tu programa una sentencia RESTORE.

Supongamos por ejemplo que quieres mirar los números de teléfono de algunos de tus amigos. La forma inmediata de hacerlo es poner en el programa una rutina de «Otra Vez». Sustituye el STOP en la línea 50 por GOTO 80, y a continuación añade las siguientes líneas de programa:

```
80 LOCATE 0,12:PRINT
   "Quieres otro numero
   (s/n)?"
90 V$=INKEY$:IF V$=""
   THEN GOTO 90
100 IF V$="s" OR V$="S"
    THEN GOTO 5
110 IF V$="n" OR V$="N"
    THEN END
```

¿Qué sucede cuando intentas ejecutar el programa? La primera vez el programa funciona perfectamente. Pero cuando pulsas una tecla para intentar ejecutarlo por segunda vez, te falla porque le faltan datos. Ello se debe a que durante la primera ejecución llegó hasta el final de la lista de datos. Afortunadamente hay una solución muy sencilla para este problema. Teclea:

```
15 RESTORE
```

Esta vez el programa funciona bien todas las veces. Ello se debe a que la instrucción RESTORE hace que el ordenador vuelva hasta el principio de la lista de datos.

Es una buena idea poner RESTORE cerca del principio del programa durante la fase de desarrollo. Esto te evita tener que recorrer todas las listas de datos en las pruebas de ejecución. Si crees que más adelante te va a convenir suprimir dicha línea de RESTORE, pon una sentencia REM con un mensaje para que te lo recuerde.

Por medio de RESTORE, puedes volver a utilizar una lista de datos tantas veces como sea necesario. Esto resulta particularmente útil en programas que tienen etiquetas, encabezamientos, tablas, etc, que serán repetidas en momentos diferentes. Por ejemplo, puedes poner los meses del año en una lista de datos dentro de una sentencia DATA en un programa

SIGUIENDOLE LA PISTA A TUS DATOS

Sea lo que sea lo que contengan los datos de tus sentencias DATA, todos ellos suelen tener una cosa en común: se requiere bastante tiempo y trabajo para calcularlos y teclearlos. Esto se aplica principalmente a la primera vez que trabajas con el programa, pero puede que también resulte necesario volver sobre el mismo mucho más adelante.

La colocación ordenada de los datos requiere algo más de trabajo al principio pero te puede ahorrar muchísimo tiempo en el futuro. La mejor organización a adoptar depende de los propios datos.

Cuando las sentencias DATA se usan para definir un bloque gráfico o algo por el estilo, organiza las líneas del programa para que se correspondan directamente con las filas del gráfico. Si los datos siguen un formato que se repite (como en los números de una lista de teléfonos, por ejemplo), organiza las líneas de programa con todas las entradas de la misma forma.

GANADORES DE LOS MEJORES DE INPUT MSX

En el sorteo correspondiente al número 5 entre quienes escribisteis mandando vuestros votos a LOS MEJORES DE INPUT han resultado ganadores:

NOMBRE	LOCALIDAD	JUEGO ELEGIDO
Carlos Revilla Puig	Barcelona	Jack the nipper
Alfonso Salinas Nieto	Barcelona	Green Beret
José Aguilar Camacho	La Carlota (Córdoba)	H.E.R.O.
Susana Munne Serrano	Barcelona	Jet Fighter
J. Antonio Tombilla Martínez	Vigo (Pontevedra)	The Dambusters
Francisco Delmás Soriano	Castellón	Billiards
Fernando Alvarez Reguant	Suria (Barcelona)	Gunfright
Miguel Palma Mur	Mequinenza (Zaragoza)	Yie ar kung fu
M.ª José Córcoles Cenón	Murcia	Bounder
Antonio Malagón Camacho	Cuenca	Yie ar kung fu

de calendario del tipo que sea, el cual sin duda hará un repetido uso de dichas tablas.

La instrucción RESTORE es especialmente útil en los casos en los que deseas seguir leyendo con READ una lista para buscar un determinado elemento, como es el caso del programa de lista telefónica que hemos visto anteriormente.

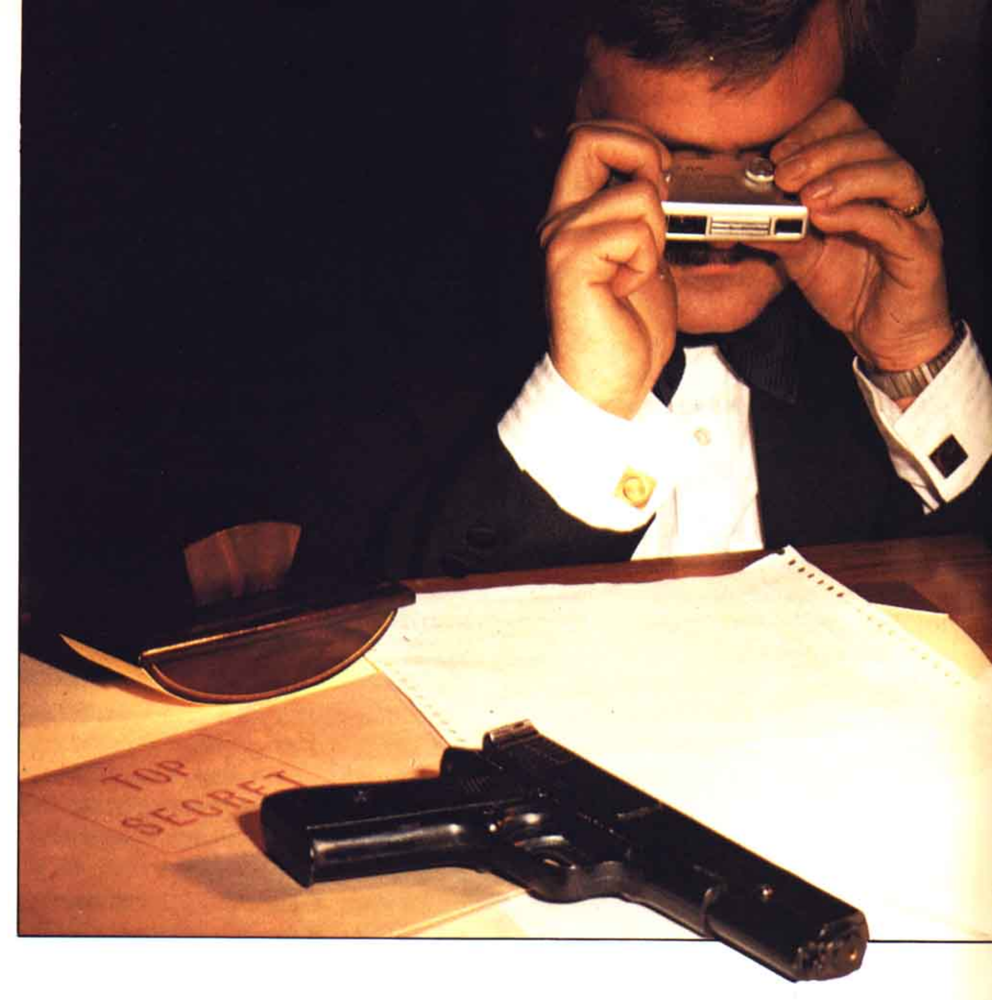
USO DE LA SENTENCIA «DATA»

Las listas de datos que contienen las sentencias DATA son extremadamente útiles en toda clase de programas. Puede que hayas hecho uso de ellas en los programas de gráficos definidos por el usuario y en el dibujo del laberinto. También hemos visto cómo puede leer el ordenador los parámetros de la música y otros efectos sonoros.

En los juegos de aventuras, se suele utilizar con frecuencia gran cantidad de líneas de datos en las que se pone todo el texto necesario. Los juegos de marcianitos escritos en BASIC también usan con frecuencia las sentencias DATA para definir personajes y otros elementos del juego.

Los programadores más expertos utilizan las sentencias DATA para hacer programas en código máquina o en lenguaje ensamblador. Tales programas pueden consistir en un corto bucle que contiene instrucciones READ, para leer los datos almacenados en las sentencias DATA, e instrucciones POKE para almacenar dichos datos en una determinada zona de memoria. Esto es lo que se conoce normalmente como un «cargador BASIC».

En una palabra, todo programa que contenga texto ordinario, números o funciones, puede hacer uso con gran provecho de las listas de datos. Con un uso cuidadoso estas listas pueden convertirse en una poderosa herramienta de programación.



USO DE LOS PUNTEROS DE «RESTORE»

El único problema que presentan las sentencias DATA es que siempre hay que llamar a la información que contienen en la misma sucesión, empezando por la primera posición. Con el uso de RESTORE siempre puedes volver al principio de la lista, incluso aunque todavía no hayas llegado hasta el final. ¿Pero cómo podrías hacer para saltar al centro de una lista?

El BASIC **MSX** te brinda una manera de hacer esto. En vez de utilizar únicamente una lista, puedes usar realmente varias, haciendo que el ordenador se dirija a la lista apropiada. En el programa de gráficos que figura más adelante, prueba a introducir las siguientes líneas adicionales:

```
6 INPUT"Puente viejo o
  moderno";Y$
7 IF Y$="v" THEN
  RESTORE 1000
8 IF Y$="m" THEN
  RESTORE 2000
9 IF Y$<>"v" AND Y$<>"m"
  THEN GOTO 5
2000 DATA 83,84,85,86,87,88,8
     9,90,90,90,90,90,90,90,9
     0,89,88,87,86,85,84,83
2010 DATA 42,55,63,65,63,55,
     42
```

Los números que figuran a continuación de los comandos RESTORE se llaman punteros de RESTORE. Dirigen al ordenador hacia una determinada lista de datos que empieza en el correspondiente número de línea. Según esto, si pulsas V para tener un puente viejo, el puntero de RESTORE toma el valor 1000.

De hecho, cuando quieras que el ordenador se dirija al primer elemento de la lista DATA, no hace falta utilizar un puntero. Si se hace un RESTORE que no va acompañado por ningún número de línea, equivale a un RESTORE con el número de la primera línea DATA. La línea 7 del anterior programa podría haberse escrito igualmente como IF Y$=«V» THEN RESTORE, obteniendo exactamente el mismo resultado.

Los punteros de RESTORE son muy útiles en la programación de juegos. Por ejemplo en un juego de aterrizajes, podrías escribir un programa para comprobar si has aterrizado con suavidad o te has estrellado. Las listas de datos podrían contener en ese caso información para emitir el correspondiente ruido de impacto o una fanfa-



rria victoriosa, y el puntero de RESTORE apuntaría en cada caso a la lista correcta.

USO DE LOS «DATA» EN LOS GRAFICOS

Las sentencias DATA son muy útiles en los programas de gráficos para fijar las coordenadas que hay que dibujar y para llamar a las rutinas gráficas de la ROM. El uso de los DATA de esta manera resulta ideal para las formas irregulares, ya que se necesitarían muchas líneas de programa si sólo se emplearan instrucciones individuales de impresión o dibujo.

No obstante existen casos en que una sentencia DATA no es de gran utilidad. Así ocurre cuando estás dibujando formas muy regulares en las que las coordenadas o el contorno gráfico pueden calcularse fácilmente.

Las sentencias DATA del programa que veremos a continuación, han sido divididas deliberadamente en varias líneas. Se ha hecho así porque la posterior modificación de los datos puede ser difícil a menos que se tengan unas divisiones muy claras; imagínatè lo que sería volver sobre este programa dentro de unos meses e intentar localizar y modificar unos cuantos datos de cada una de las variables. Naturalmente, puedes seguir detenidamente la pista de cada una de las instrucciones READ, pero hacer esto en un programa largo es muy tedioso. Por ello te recomendamos que fragmentes cada grupo de sentencias DATA en varias líneas y que además (suponiendo que tengas memoria disponible) utilices abundantemente las sentencias REM para que las cosas te resulten más claras.

El siguiente programa utiliza READ y DATA para ayudarte a dibujar un puente. Si introduces y ejecutas el programa por etapas, resultará más fácil comprender lo que sucede y comprobar que no te has equivocado al teclear. Empieza con lo siguiente:

```
10 SCREEN2:FOR T=94 TO
   104 STEP 5
20 PSET(35,T)
30 CIRCLE STEP(87,0),87,,0,
   3.1415
40 NEXT T
```

Esta sección utiliza un bucle FOR ... NEXT que permite el dibujo de tres puntos próximos al lado izquierdo de la pantalla y a continuación dibuja una línea curva a partir de cada uno de dichos puntos. Esta línea curva es un arco de circunferencia que se obtiene con la función CIRCLE. El centro se sitúa mediante STEP a 87 *pixels* a la derecha de cada uno de los puntos. La semicircunferencia se consigue especificando los ángulos inicial y final como 0 y PI (3.1415).

```
100 FOR N=18 TO 39
110 READ A
120 PSET (N,110)
130 LINE -STEP(0,-A)
132 PSET (N+188,110)
134 LINE -STEP(0,-A)
140 NEXT N
1000 DATA 70,70,67,67,70,70,6
     0,60,57,57,60,60,57,57,6
     0,60,70,70,67,67,70,70
```

Esta sección es la que dibuja las torres. El bucle situado entre las líneas 100 y 140, más el número 110 (*pixels* contados a partir del fondo) de las líneas 120 y 132 dibuja los puntos de la base de cada una de las líneas verticales muy próximas que forman las torres.

Seguidamente entran en acción las líneas 110 y 1000. La línea 110 dice al ordenador que lea en la línea 1000 la altura, expresada en *pixels*, de cada una de las 22 líneas verticales. Así, la primera línea es 0,70 es decir, vertical y de una altura de 70 *pixels*; la segunda línea es 0,70, la tercera 0,67 y así sucesivamente. Si quieres ver cómo va sucediendo esto, ensaya a insertar temporalmente una instrucción como:

```
135 FOR PA=1 TO 100:NEXT
300 PSET (0,75)
310 LINE -STEP(255,0)
320 PSET (0,78)
330 LINE -STEP(255,0)
```

Estas líneas son las que se ocupan de dibujar la carretera. Por último:

```
400 FOR R=62 TO 182
    STEP 20
410 PSET (R,78)
420 READ B
430 LINE -STEP(0,-B)
440 NEXT R
1010 DATA 42,55,63,65,
     63,55,42
```

Estas líneas se encargan de dibujar los cables verticales que unen el arco con la carretera. El bucle FOR ... NEXT ayuda a dibujar las posiciones de partida de los cables, mientras que las líneas 420 y 1010 controlan su altura.

Si quieres que el programa vuelva a ejecutarse automáticamente, añádele las siguientes líneas:

```
5 CLS:RESTORE
450 FOR J=1 TO 1000
    :NEXT:GOTO 5
```

La línea 5 borra la pantalla, permitiendo luego que el programa lea de nuevo los datos.

LOGO MSX: EL LENGUAJE DE LA TORTUGA

- UN LOGO EN CASTELLANO
- LAS TORTUGAS Y SUS FORMAS
- DIBUJANDO CON LA TORTUGA
- PROCEDIMIENTOS EN LOGO
- FUNCIONES MATEMATICAS

El LOGO nació como un lenguaje de programación altamente intuitivo, estructurado de una forma muy lógica y asequible, para aquellos cuyo idioma nativo es el inglés.

Afortunadamente existe ya una versión española del LOGO MSX y ahora si podemos aprovecharnos de todas las ventajas que el lenguaje nos ofrece.

Aún cuando evidentemente no varía la estructura, está claro que resulta más expeditivo ordenar desaparecer la tortuga con «esconde la tortuga» que con «*hide turtle*» o dejar una traza con «baja lápiz» que con «*pendown*».

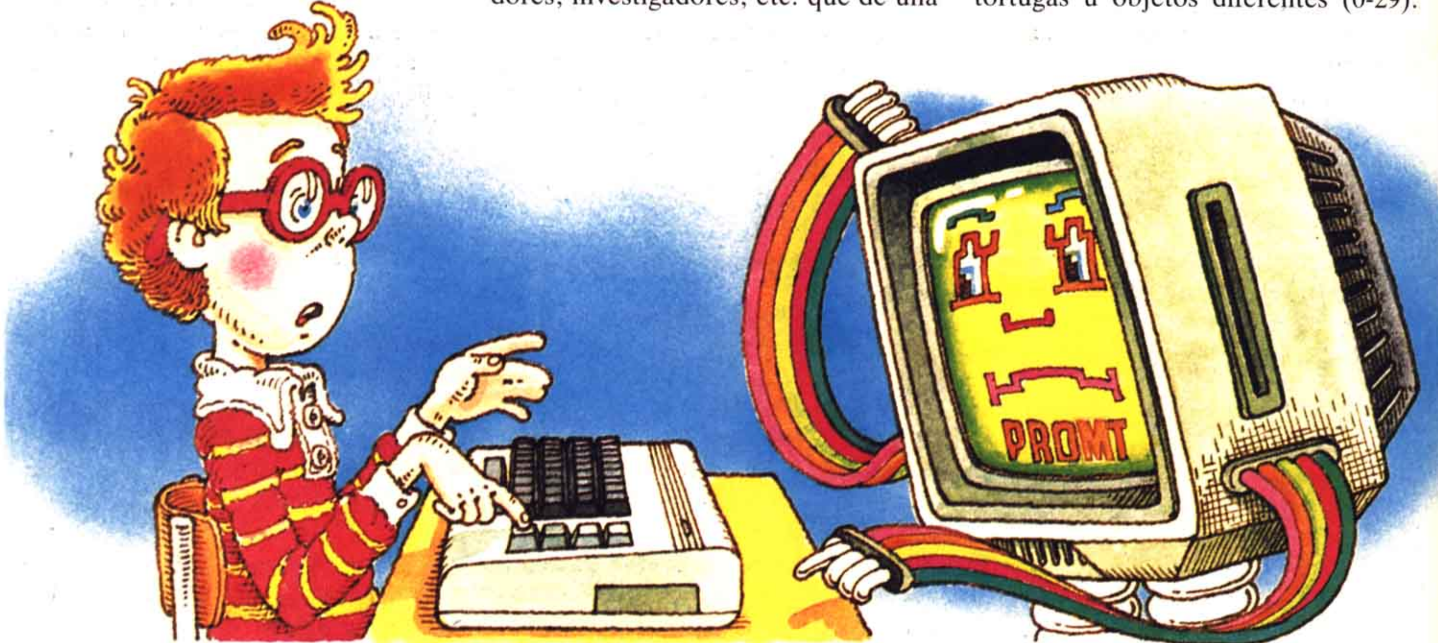
No vamos a contar todo aquello que afecta al LOGO de una manera general, entre otras cosas porque en el número 1 de esta misma revista se publicó el artículo LOGO, LENGUAJE INTUITIVO donde el lector hallará una excelente introducción y una presentación general de este interesante lenguaje. Nos centraremos exclusivamente en aquellos aspectos del LOGO **MSX** español que consideremos de mayor interés inmediato.

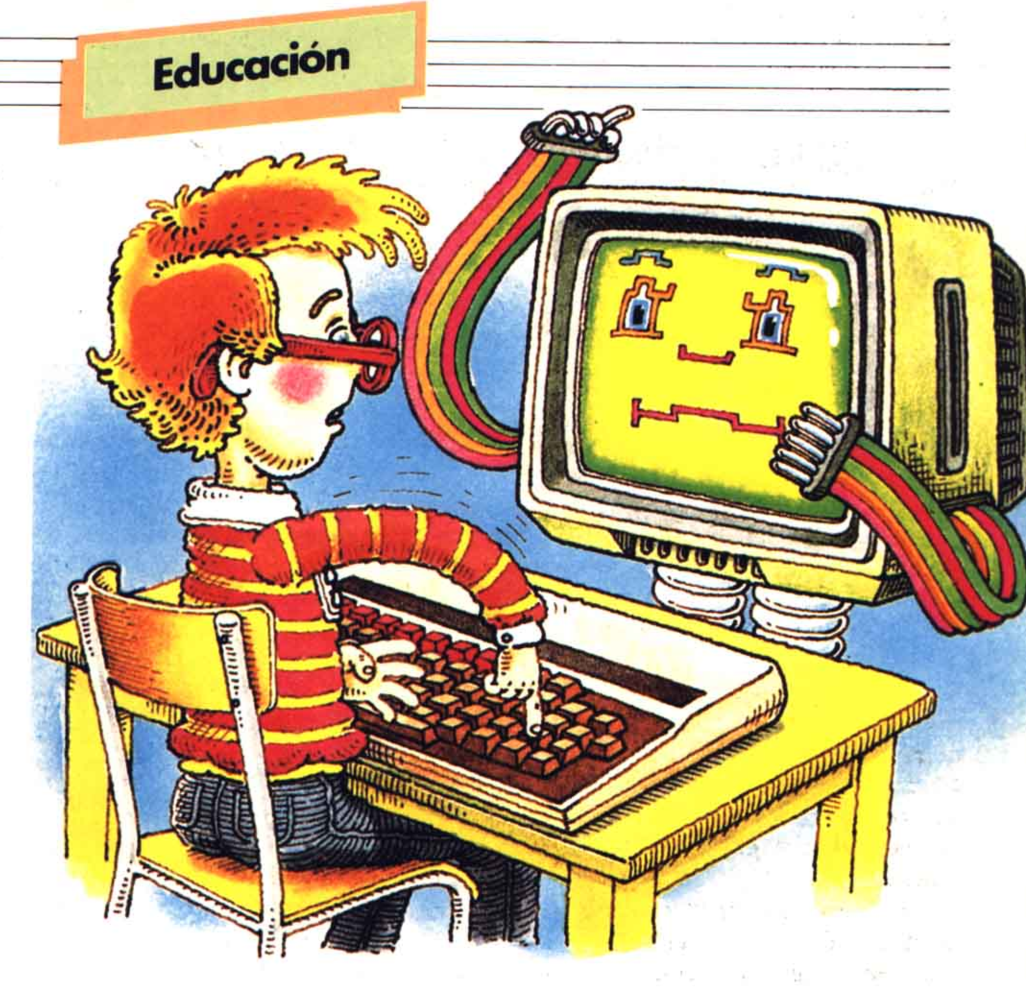
Antes de continuar conviene aclarar algunos aspectos para evitar malentendidos. Al LOGO se le asocia generalmente a las tortugas, a los gráficos, a los niños, y a la ausencia de conocimientos de programación. Es verdad, pero no toda la verdad. Evidentemente es un lenguaje tremendamente lógico, podríamos decir que uno de los más «humanos» de entre los lenguajes existentes, gracias al cual se puede acometer la tarea de controlar un ordenador para hacer ciertas funciones con un mínimo de conocimientos y de esfuerzo... hasta los niños pueden hacerlo. Ciertamente, para un niño resulta tan fácil hacer dibujos sencillos, jugar con colores o hacer algunas cuentas con ordenador que con papel, lápiz, goma de borrar y pinturas de colores. Pero también hay que admitir que cuando se trata de hacer algo más sofisticado, la candidez del sistema va desapareciendo. Con todo, el LOGO es un lenguaje altamente formativo para todas aquellas personas, educadores, investigadores, etc. que de una u otra forma deban utilizar el ordenador como complemento o sustitución de actividades humanas como pueden ser la robótica, inteligencia artificial, o simplemente como mentalización informática.

LAS TORTUGAS Y SUS FORMAS

En el horizonte del LOGO existen una serie de «mundos», de los cuales el de los gráficos es sin lugar a duda el más importante, y dentro de él, el «mundo de la tortuga». En el caso del LOGO **MSX**, y algún otro, debemos hablar mejor que de tortuga, de «comodín» ya que ésta puede tomar la forma de un helicóptero, un perro, una pared de ladrillos o un cohete. Pueden utilizarse hasta 60 figuras o formas (0-59), de las cuales 34 están predefinidas al encender el equipo. Todas ellas pueden ser rediseñadas por el programador.

La facilidad del **MSX** para manejar *sprites* se refleja también en el LOGO **MSX**. Se pueden controlar hasta 30 tortugas u objetos diferentes (0-29).





El control se ejerce mediante la primitiva «designa» seguida del número de una tortuga en concreto, una lista de números de tortugas o la primitiva «todo» para cubrir las 30 posibles tortugas a la vez.

Resumiendo, podemos manejar a la vez de 1 a 30 tortugas u objetos, y para cada una podemos elegir de entre 60 formas. Cada tortuga está siempre identificada por un «número de orden» y un «número de forma». Por razones históricas se sigue dando el nombre genérico de «tortuga» a cualquiera de los posibles 30 objetos. Esto puede crear inicialmente alguna confusión. Por ejemplo, en el caso de que funcionemos con la tortuga 0 con forma de gato (2) y la tortuga 4 con forma de auténtica tortuga (36) puede resultar confuso dar la orden de que avance la tortuga 0 20 pasos y en la pantalla se mantenga estática la única tortuga visible y por el contrario se desplace el gato...

Por el momento, para no complicar las cosas, nos vamos a limitar a una sola tortuga, aunque, más adelante jugaremos con varias a la vez.

La forma de la tortuga se asigna mediante la primitiva «ffigura<número (0 - 59)>».

En un momento dado podemos saber la forma correspondiente a la tortuga en curso mediante «figura». Con ayuda de «quien» podremos conocer con qué número de tortuga estamos operando.

Con la primitiva «rg» hacemos una restauración de las tortugas a su estado inicial y con «vp» borramos el texto que hubiera en la pantalla. Con «es» (escribe) podemos reflejar en pantalla cualquier resultado. Se trata del equivalente a la instrucción PRINT del BASIC.

```
?ffigura 8  tortuga/locomoto
                          ra
?ffigura 4  trotuga/camion
?ffigura 38 tortuga/tortuga
                        girada
?es figura  38
?es quien   0
?rg designa 2
?es figura  36
?es quien   2
```

DISEÑANDO FORMAS DE TORTUGA

Diseñar la forma de una nueva tortuga o rediseñar cualquiera de las existentes resulta bastante simple con el **Editor de Figuras** del LOGO **MSX**. Recordemos que inicialmente las figuras de 10 a 35 están sin definir.

La base de cualquier figura es un cuadrado compuesto a su vez de 16 × 16 cuadrados o casillas elementales, que pueden dejarse vacías o rellenarse de un color determinado (el mismo para todas). Podremos cambiar de color posteriormente, mientras utilizamos la tortuga.

Supongamos que deseamos hacer una «cara sonriente» para utilizarla posteriormente en un programa y queremos asignarle el número 12. En primer lugar limpiaremos la pantalla «vp» y a continuación entraremos en el editor de figuras mediante «edfi 12». Si el número 12 corresponde a una figura aún no definida aparecerá en el centro de la pantalla un gran cuadrado macizo (con CLR HOME aparecerá el reticulado de 16 × 16). En caso contrario aparecerá la figura predefinida. Si queremos que no aparezca la tortuga en curso en el centro podemos eliminarla con «et», aún cuando no afectará nuestro diseño en caso de que decidamos dejarla donde está.

Con ayuda de las cuatro teclas de dirección podemos situar el cursor parpadeante donde nos interese y mediante la barra espaciadora conseguiremos que la casilla correspondiente se llene de color o se vacíe.

Si consideramos que la «cara sonriente» está a nuestro gusto pulsaremos ESC y habremos aumentado nuestro catálogo de formas de tortugas. Si no nos gusta podemos retocarla mediante el procedimiento descrito o incluso borrarla entera para comenzar de nuevo (CTRL-K ó CLS/HOME).

Si lo que realmente queremos es abandonar la idea, sin que quede rastro de nuestro intento, CTRL-STOP lo hará por nosotros.

Tal vez nos hayamos metido a modificar una figura ya existnte pero al final no nos gusta lo que hemos hecho, por aquello de más vale lo malo conocido... Tampoco tenemos por qué

preocuparnos, CTRL-Y nos restaura la figura inicial de la tortuga.

Si queremos hacer criaturas clónicas o duplicadas, podemos ayudarnos de la primitiva «copiafi <número de la figura><número nueva figura>». Esto puede resultar útil en nuestro caso si queremos aprovechar la «cara sonriente» número 12 para confeccionar una «cara triste» como número 14. Para ello lo primero que tenemos que hacer es copiar la cara sonriente asignándola a la figura 14. Después pasaremos a editar. Todo esto se consigue con las siguientes líneas:

```
?ffigura 12   cara sonriente
?ffigura 14   cuadro en blanco
?copiafi 12 14
?ffigura 14   cara sonriente
?edfi 14      para modificar
              la figura
```

Si apagamos el ordenador o tenemos que hacer un RESET perderemos todo el trabajo si no hemos tenido la precaución de archivarlo en *cassette* o disco (guardac <nombre> ó guarda <nombre> respectivamente).

Como ejercicio podemos visualizar una a una, de una manera automática, las 60 formas de las tortugas que tenemos programadas en un momento dado. Basta con que tecleemos:

```
?vp rg
 ffigura 0 fcursor [10 12] es
 figura espera 50
 repite 59
 [ffigura figura + 1 fcursor
 [10 12] es figura espera 50
 vp rg]
```

La composición, byte a byte, de una figura se obtiene con «obfi <número>». Los 32 valores obtenidos se corresponden con la siguiente distribución: byte superior de la mitad izquierda del *sprite*, ..., byte inferior de la parte izquierda, byte superior de la parte derecha,..., byte inferior de la parte derecha.

En el caso de la cara sonriente número 12 la composición es la siguiente (obtenida con «es obfi 12»):

DIBUJO CON LA TORTUGA

Se supone que todas las tortugas llevan debajo de su caparazón un lapicero que pueden bajar («bl») para pintar cuando se mueven o levantar («al») para no marcar durante el desplazamiento. También van provistas de una goma que utilizan para borrar («gl») los trazos que encuentran al pasar, cuando conviene.

El desplazamiento de las tortugas puede hacerse avanzando («av») o retrocediendo («re») un número cualquiera de pasos, o girando a la derecha («de») o izquierda («iz») un número de grados estipulado. Mediante «centro» podemos hacer que en un determinado momento la tortuga deje de hacer lo que está haciendo y vuelva «a casa» (posición central de la pantalla).

También puede fijarse el rumbo (0-360 grados) mediante «frumbo <grados>» o conocerse éste mediante «rumbo».

Las siguientes líneas son un ejemplo de todo esto:

```
?rg vp al av 50 avanzar 50
                pasos
?de 90 av 50    girar derecha
                y avanzar 50
?centro         retorna tor-
                tuga a posi-
                cion inicial
?frumbo 45      80 pasos en
 av 80          direccion nor
                oeste
?centro repite  restaura tor-
 15 [al av 2 bl tuga y traza
 av 2 espera    10 lineas dis
 20]            continuas
```

```
fila   decimal    binario             grafico
----   -------    ----------------    ----------------
01     015 240    0000111111110000        ********
02     016 008    0001000000001000       *        *
03     032 004    0010000000000100      *          *
04     064 002    0100000000000010     *            *
05     134 097    1000011001100001    *    **  **    *
06     134 097    1000011001100001    *    **  **    *
07     128 001    1000000000000001    *              *
08     129 129    1000000110000001    *      **      *
09     129 129    1000000110000001    *      **      *
10     128 001    1000000000000001    *              *
11     136 017    1000100000010001    *   *      *   *
12     132 033    1000010000100001    *    *    *    *
13     067 194    0100001001000010     *    *  *    *
14     032 004    0010000110000100      *    **    *
15     016 008    0001000000001000       *        *
16     015 240    0000111111110000        ********
```

La tortuga podemos hacerla aparecer («mt») o desaparecer («et»). En el caso anterior podíamos haber conseguido el trazado discontínuo sin que se viera la tortuga, añadiendo «et» después de «centro».

La primitiva «limpia» elimina los gráficos que haya en la pantalla pero sin afectar a las tortugas.

Por lo que respecta al color podemos elegirlo libremente tanto para la tortuga («fct») como para el lapicero («fcl») o el fondo de pantalla («ffo»). Las primitivas «color», «cl» y «fondo» nos informan respectivamente de los colores de la tortuga, lapicero y fondo que estamos utilizando.

El color 0 (transparente) hace desaparecer lo que se pinte con él. Los quince colores que pueden utilizarse (1-15) son respectivamente: negro, verde medio, verde claro, azul oscuro, azul claro, rojo oscuro, lila, rojo medio, rojo claro, amarillo, amarillo claro, verde oscuro, magenta, gris y blanco.

Supongamos que gracias al lapicero

CURSO DE INGLES

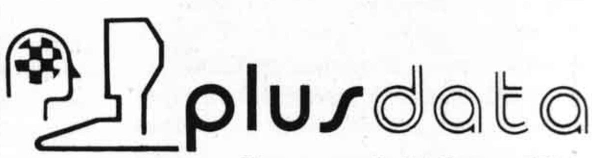
The Gruneberg Linkword Language System es un sistema, para enseñanza de idiomas, más rápido y fácil que los métodos convencionales aplicados actualmente.
En poco tiempo, máximo 20 horas, te enseñará un vocabulario de 400 palabras y adquirirás unas buenas nociones de gramática. Esto te permitirá entender y ser entendido en tus viajes a lugares de habla inglesa o en tus contactos con personas que se expresen en ese idioma.
Por otra parte, el Sistema PlusData, consigue que el ordenador se convierta en un perfecto profesor que te explicará, orientará y corregirá, manteniendo en todo momento un "diálogo" interactivo de resultados sorprendentes.



–L. Taylor. "POPULAR COMPUTER WORLD":
"Quedé francamente atónito al comprobar la efectividad de la sugestión de imágenes como elemento de ayuda a la retención..."

–"PERSONAL COMPUTER WORLD":
"Un suceso fuera de serie..."

–Bill Barnet. "COMPUTER CHOICE":
"De todos los paquetes para aprender idiomas éste es el más interesante..."



Programas de EAO para EGB.
Cursos de Basic, Cobol, etc. AUTODIDACTAS.

Nombre ....
Apellidos ....
Dirección ....
Población ....
D.P. .... Tlno. ....
Forma de pago: Reembolso ☐ Giro postal ☐ Envío talón ☐

☐ Curso de Inglés 1.ª parte. 10 lecciones Linkword. (Cinta) P.V.P. 6.900.-Ptas.
☐ Curso de Inglés 1.ª parte. 10 lecciones Linkword. (3,5"-Disk) P.V.P. 7.900.-Ptas.
ENVIAR ESTE CUPON A: PLUS DATA, S.A. C/. GRAN VIA, 661 pral. 08010-Barcelona. Tel. 246 02 02

de la tortuga hemos dibujado un triángulo, que a continuación queremos pintar del mismo color. Basta con que nos situemos en un punto cualquiera del triángulo y utilicemos «bl llena».

Por último podemos dibujar un punto en la pantalla con «punto [x y]».

Las tortugas pueden situarse en una determinada parte de la pantalla mediante «fpos [x y]» donde X e Y son las coordenadas del lugar deseado. Pueden situarse en otra posición del eje X mediante «fijax X» o en el eje Y con «fijay Y» si sólo se pretende desplazarse según uno de los ejes. La posición de la tortuga puede conocerse mediante «pos», «coorx» ó «coory» respectivamente.

Además de la posibilidad de avanzar paso a paso se puede hacer con gran facilidad que las figuras se desplacen a una cierta velocidad («fvel») e incluso ésta puede fijarse independientemente en sentido horizontal («fvelx») o vertical («fvely»). Los correspondientes valores pueden conocerse mediante «vel», «velx» y «vely».

MANEJO DIRECTO DE COMANDOS

Lo dejaremos enteramente a criterio del lector, de acuerdo con su talento creativo, pero sugeriremos ir probando uno a uno todos aquellos comandos o primitivas tales como avanzar, retroceder, cambiar colores, borrar, posicionar, girar etc. Si no tenemos a mano el manual podemos recordar la lista de las primitivas mediante «primitivas» (STOP para parar y cualquier tecla para seguir). Teclea el siguiente ejemplo:

```
?fpos [-10 -10]
?bl ffigura 5 av 20
?fcl 8 de 30 av 30 re 10 de
 45 av 30
?et iz 90 av 15
?mt al av 10 ffo 5 fct 15
```

PROCEDIMIENTOS

Mediante comandos directos, uno a uno, podemos hacer muchas cosas, pero con gran lentitud y esfuerzo, sobre todo cuando existen procesos repetitivos.

Con lo que hemos visto hasta ahora podríamos hacer un programa para dibujar una casa. Lo malo es que si quisieramos hacer otra distinta deberíamos partir también de cero, aunque utilicemos elementos comunes. Es fácil de imaginar que si tuvieramos procedimientos para dibujar ventanas, puertas o tejados, programar el diseño de una casa sería, como utilizar piezas de un «mecano».

«Procedimientos» en LOGO, son programas auxiliares o «recetas» para efectuar funciones concretas (gráficos, manejo de tortugas, procesos matemáticos, musicales, etc.). Una vez generados, los «procedimientos» pueden utilizarse tantas veces como se quiera, incluso en forma recursiva. La mecánica para generar procedimientos es como sigue:

```
?para <nombre>[:var1 :var2 ]
 .......
 lista de comandos
 .......
 fin
```

Para ejecutar un procedimiento basta llamarle por su nombre, seguido de las variables correspondientes (si se utilizan). Más adelante veremos una serie de ejemplos.

Para conocer los «procedimientos» existentes en memoria podemos utilizar «imts». Si deseamos eliminar uno en especial utilizaremos «borra»<nombre> pero si lo que deseamos es ver su composición y quizá modificarlo deberemos emplear «edita»<nombre>.

ESPIRALES Y FIGURAS GIRATORIAS

```
A.
?para espiral :paso :angulo
 av :paso
 de :angulo
 espiral :paso+2 :angulo
 fin
```

Acabamos de generar un procedimiento con dos variables (paso y ángulo). Aunque no hemos manejado aún los procedimientos no creemos que éste resulte difícil de comprender, y además resulta muy ilustrativo.

Previa limpieza de pantalla podemos probar diversos valores de «paso» y «ángulo» hasta conseguir figuras interesantes. Prueba por ejemplo:

```
B.
?para cuadrado
 bl et
 repite 4[av 30 de 90]
 fin
?repite 8[cuadrado de 45]
```

Genera un cuadrado giratorio alrededor del punto central de la pantalla. Prueba otros valores y añade color.

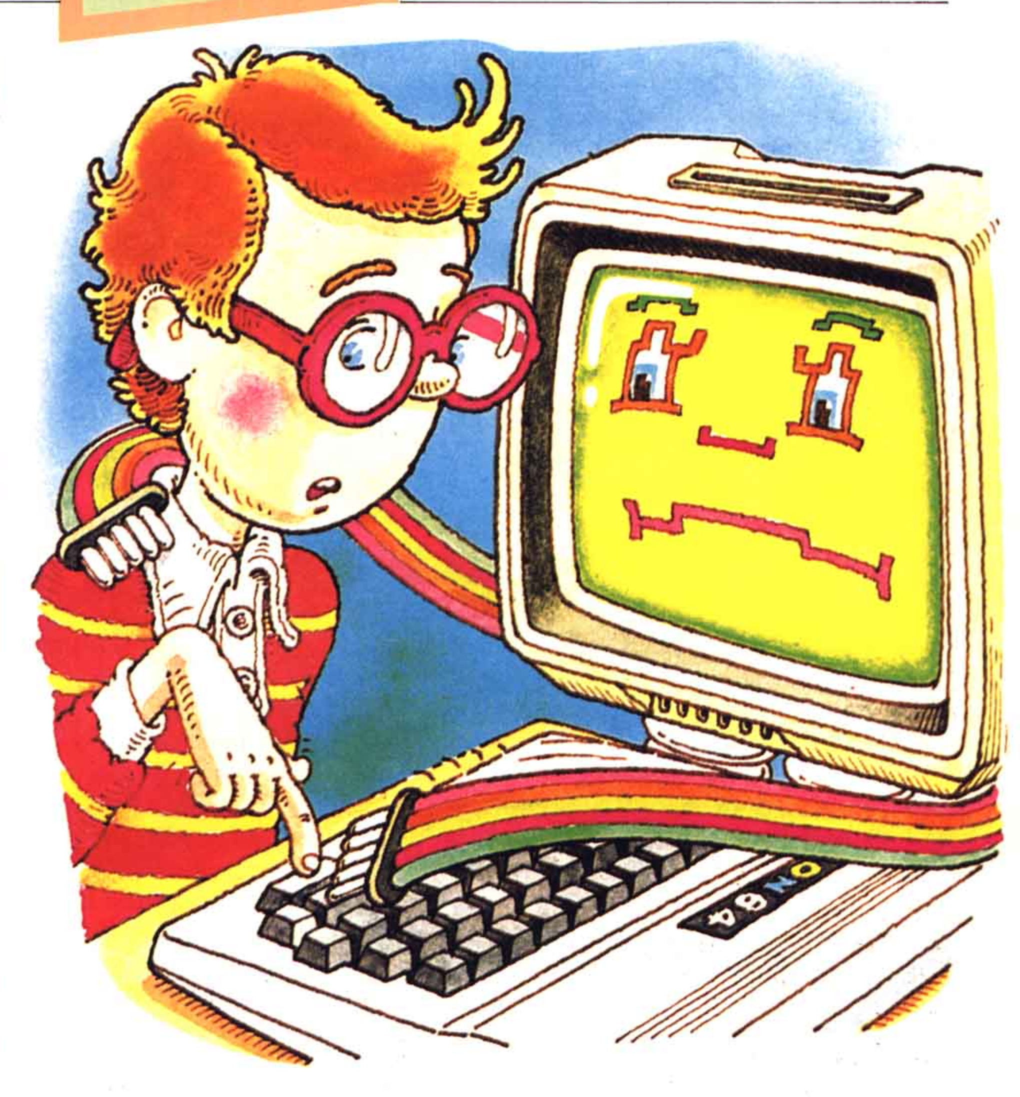
```
C.
?para rectangulo
 bl et
 av 30 de 90
 av 50 de 90
 av 30 de 90
 av 50
 fin
?repite 15[rectangulo de 15]
```

Igual que B pero utilizando rectángulos

MOVIMIENTO

```
D.
?para via
 fijax -120
 et de 90
 bl av 240
 fijay -2
 re 240
 fin
?para tren
 rg vp via
 fijay 3
 ffigura 8 mt al
 fvelx 3
 fin
?tren
```

Este sencillo listado dibuja una vía y hace que circule una máquina de izquierda a derecha a una cierta velocidad (3). Probar otras velocidades y otras figuras, por ejemplo la número 3.



PINTURA DE SUPERFICIES

```
E.
?para colores
 rg vp cuadrado
 frumbo 45
 et al et av 10 bl
 repite 14 [fcl cl-1 llena
 espera 30]
 fin
?colores
```

Estas líneas trazan un cuadrado (el diseñado para el ejercicio A) y lo van llenando de diversos colores.

LAS MATEMATICAS Y EL LOGO MSX

Aunque el LOGO no es un lenguaje pensado para cálculos matemáticos complicados, sí nos permite manejar con gran facilidad una serie de comandos básicos como son las cuatro reglas («suma»< A B >, «dif»< A B >, «producto» < A B >, «cociente»< A B >), funciones trigonométricas (sen, cos, arctan), raíz cuadrada («rc») y una serie de funciones útiles como son el redondeo («redondea»), parte entera («ent»), generación de números aleatorios («azar») y operaciones lógicas. Por ejemplo:

```
operacion          resultado
------------       ----------
?es ent 45,6        45
?es rc  45,6        6,757772
?es suma 7 3        10
?es seno 30         0,5
?es rc 47+32*3      11,95826
?es 27<3            falso
```

Algunas de las operaciones podemos escribirlas de una forma más habitual como por ejemplo:

```
?es 47*3            141
?es 12+3-18         -3
```

TORTUGAS MULTIPLES Y FIGURAS ANIMADAS

Las diversas tortugas puede manejarse una a una o un grupo a la vez. Para ello, antes de dar las instrucciones pertinentes debemos poner «a la escucha» la tortuga o tortugas de que se trate mediante «designa»<número de tortuga> o [lista de tortugas]. Aquellas tortugas que estén realizando alguna actividad no se ven afectadas por las instrucciones que se den a las restantes.

Como ejemplo indicamos un proceso muy elemental para hacer que tres tortugas tengan un movimiento uniforme y simultáneo de izquierda a derecha. Dejamos al lector que lo modifique para imprimir a cada objeto una velocidad distinta, que comiencen desde lugares diferentes, etc.

```
?para carrera
 rg vp
 designa 0 ffigura 3 al
 fpos[-120 30] mt
 designa 1 ffigura 4 al
fpos[-120 0 ] mt
designa 2 ffigura 7 al
fpos[-120 -30] mt
desinga [0 1 2] fvelx 10
fin
```

Para comenzar la carrera se teclea «carrera RETURN». Si durante este proceso deseamos que aparezca en escena un cohete que se desplace verticalmente en el centro de la pantalla basta con teclear lo siguiente:

```
?designa 4 ffigura 5 mt
 fvel 3
```

Tal vez nos interese detectar la colisión del cohete y el helicóptero y montar algún efecto especial. Con LOGO **MSX** español disponemos de «demonios» o «vigilantes» que nos avisan cuando se produce una colisión entre dos objetos determinados y a los cuales podemos dar una lista de ins-

trucciones a ejecutar cuando esto ocurra («en.contacto»<tortuga x> <tortuga y> [lista de instrucciones]

Estos vigilantes pueden aprovecharse también para detectar ciertas acciones como puede ser la actuación de un disparador de *joystick* («cuando»).

La acción de los vigilantes o demonios se cancela mediante «bods» (borra demonios).

Otro efecto interesante es la animación de figuras (vuelo de pájaro, carrera de un perro, etc.). Para ello puede asignarse en una secuencia y cadencia determinadas varias formas a una misma tortuga (por ejemplo diversar posturas del cuerpo en la carrera), completarse con movimientos independientes de otras tortugas (coches que pasan, objetos que aparecen, etc) y tal vez acompañarse de combinaciones de color y sonido.

Evidentemente no tenemos espacio para tocar todos los aspectos del LOGO, pero esperamos haber conseguido interesar a alguno de nuestros lectores lo suficiente como para justifcarles trabajar pacientemente el ordenador con ayuda del manual de instrucciones y llegar a dominar este lenguaje de programación que se está haciendo tan popular, fundamentalmente en el campo de la enseñanza.

EL ZOCO DE INPUT



Todo se compra y se vende. Los antiguos zocos fueron lugares destinados a todo tipo de transacciones. INPUT también tiene el suyo. Vuestras operaciones de compra, cambio o venta serán publicadas en esta sección, pero dos son las limitaciones que imponemos:

a) La propuesta tendrá que ver con la microinformática.

b) Nos reservamos el derecho de no publicar aquellos insertos de los que se sospeche un trasfondo lucrativo.

Ahora un ruego. Tratar de resumir al máximo el texto; escribir casi como un telegrama siendo claros y concisos.

Envía tu mensaje a:

INPUT MSX ZOCO
P.º de la Castellana, 93. Planta 14
28046 MADRID



¡NO TE PIERDAS NI UN SOLO EJEMPLAR!

INPUT MSX quiere proporcionar a sus lectores este nuevo servicio de ejemplares atrasados para que no pierdan la oportunidad de tener en sus hogares todos los ejemplares de esta revista, líder en el mercado español.

Podréis solicitar cualquier número de **INPUT MSX** que querais, siempre al precio de cubierta (sin más gastos).

Utiliza el cupón adjunto, enviándolo a **EDISA** (Dpto. de Suscripciones), López de Hoyos, 141 - 28002 Madrid, o bien llámanos por teléfono al (91) 415 97 12.





CUPON DE PEDIDO

SI, envíenme contrarreembolso .... ejemplares de **INPUT MSX** de los números:

(marca con una (X) tu elección)

[1] [2] [3] [4] [5]

NOMBRE

APELLIDOS

DOMICILIO

NUM. PISO ESCALERA COD. POSTAL

POBLACION PROV.

TELEFONO FIRMA

UN PROGRAMA QUE APRENDE

Con este artículo, vamos a tratar de introduciros en el apasionante mundo de la Inteligencia Artificial, prescindiendo de las largas y aburridas parrafadas teóricas habituales en este tipo de trabajos.

Para que ello sea posible, debemos dar por supuestos algunos principios muy básicos que, a pesar de su sencillez, es posible que algún lector no conozca. A los que se encuentren en ese caso, les remitimos al manual de uso de su «micro», o a cualquier libro sobre programación en BASIC **MSX**.

En cuanto a aquellos que disfruten de un nivel más avanzado, les invitamos a olvidarse del texto de este artículo y a centrarse en el estudio del programa que lo acompaña el «programa inteligente» que hemos llamado **Sabelotodo**.

A pesar de estar escrito en BASIC, muestra lo que la **Inteligencia Artificial** puede llegar a hacer a través de un breve puñado de instrucciones. Otra ventaja, además de su corto listado, es que está pensado para que el lector lo modifique según sus necesidades o aficiones.

SABELOTODO

Se trata de un curioso ejemplo de aplicación de la **Inteligencia Artificial** que burla las limitaciones del BASIC e inaugura una serie de programas similares que irán apareciendo dentro de esta sección en los próximos números.

El **Sabelotodo** se basa en un sistema matricial de tablas de propiedades con **recursividad**, que permite memorizar datos y tratarlos deductivamente para «aprender». Es decir, un programa ¡que aprende!

Lo hemos hecho de forma que lo que aprenda sea Historia, pero es susceptible de ser modificado, con unos pocos retoques, para trabajar de la misma manera con otros temas, como Deportes, Records, etc.

A continuación pasamos a ver detenidamente qué es lo que hace, para describir después cómo lo hace:

1º.- Pensamos un personaje histórico cualquiera, del que conozcamos algunas características, comenzamos a jugar.

2º.- El programa, que ha de adivinar el nombre del personaje, nos pregunta si posee una característica concreta (alguna que ya tenga en su memoria), y así sucesivamente hasta deducir, por eliminación, cuál es el personaje que habíamos pensado.

3º.- Si el programa no tiene almacenado previamente (si el programa no ha «aprendido» previamente) el nombre y las características del personaje, se da por vencido, nos pide su nombre, y nos pregunta más cosas sobre él para ampliar sus conocimientos.

4º.- De esta forma, partiendo de un solo personaje almacenado en la memoria, puede llegar a conocer un sinnúmero de personajes, hasta llenar totalmente la memoria.

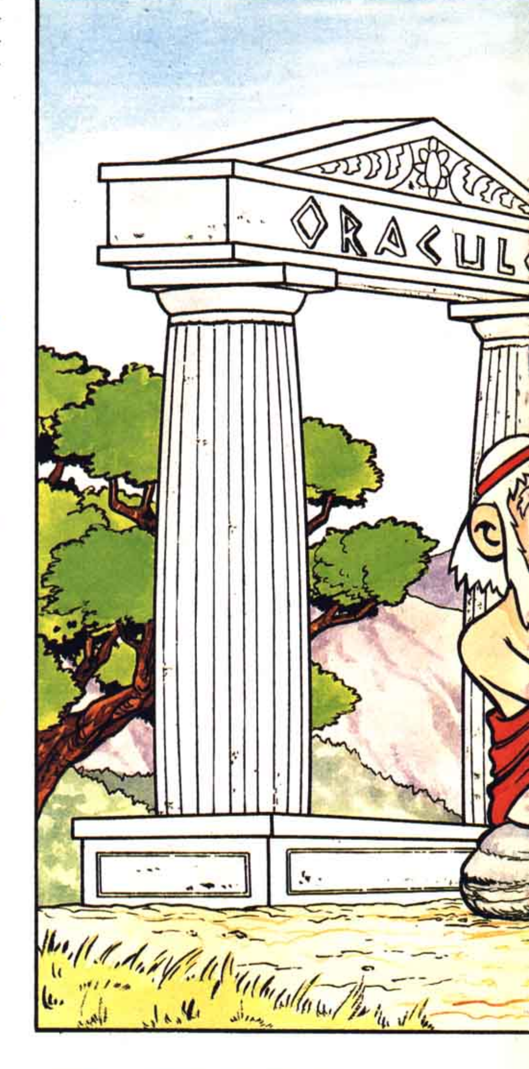
El programa está estructurado con mucha claridad y acompañado de las oportunas sentencias REM, pués su misión es precisamente la de serviros como material de trabajo para que aprendáis a hacer vuestros propios programas de **Inteligencia Artificial**.

Ahora que sabemos qué es lo que hace, vamos a ver el cómo:

1º.- El programa toma la primera característica del primer personaje almacenado en la memoria y pregunta si nuestro personaje la posee.

2º.- Si la respuesta es negativa, coloca el valor «cero» a todos los personajes que posean esa característica y pasa al siguiente que tenga todavía el valor «uno».

3º.- Si es positiva, coloca el valor



«cero» a todos los personajes que no la posean y pasa a la siguiente característica del primer personaje.

4º.- Sigue con el proceso anterior hasta que todos los personajes que conoce tengan el valor «cero», en cuyo caso se da por vencido, o hasta que encuentre uno que cumpla todas las condiciones, es decir, que hayamos respondido afirmativamente a todas sus características.

5º.- Si el personaje propuesto por el ordenador no es el que habíamos pensado (a pesar de que hubieran coincidido todas las características), o si el ordenador se da por vencido, nos pide el nombre del personaje y almacena bajo él todas las características deducidas de las preguntas y respuestas anteriores, más las que introduzcamos directamente cuando nos pida más datos.

Por si esta descripción pudiera parecer demasiado enrevesada, de hecho lo es, vamos a verlo otra vez con un ejemplo práctico. Supongamos que el



programa ya tiene almacenados en la memoria los nombres de cuatro insignes personajes históricos con sus correspondientes características, y que nosotros nos disponemos a jugar con él para que trate de adivinar un nuevo personaje que se nos ocurra. En este momento, las tablas de propiedades del programa muestran los siguientes datos:

CERVANTES
fué literato
escribió El Quijote
luchó en Lepanto
era manco

NAPOLEON
fué emperador
conquistó Europa
venció en Jena
fué desterrado

GOYA
fué pintor
trabajó en la corte
murió en 1828
era aragonés

KANT
fué filósofo
era prusiano
nació en 1724
renovó la filosofía

Nosotros pensamos un personaje cualquiera, un pintor, por ejemplo: **Velázquez**.

El ordenador comienza por preguntarnos: ¿Fué literato? (primera característica del primer personaje). Contestamos que no. A continuación el programa busca a todos los personajes que cumplan esa condición (de los cuatro, sólo la cumple **Cervantes**) y les coloca un «cero». El siguiente personaje que aún no ha sido eliminado es **Napoleón**. Por tanto, nos pregunta ¿Fué emperador? La respuesta es no, y el ordenador procede de la misma manera. El siguiente es **Goya**. El programa nos pregunta ¿Fué pintor? Contestamos que si, con lo que se coloca un «cero» para **Kant**, que no cumple esa condición. Ya sólo nos queda **Goya**. La pregunta siguiente es ¿Trabajó en la corte? Como **Velázquez** también pintó en la corte, contestamos que sí. Después preguntará ¿Murió en 1828? A lo que responderemos negativamente. El programa coloca un «cero» a **Goya** y, como ya no quedan más personajes, se da por vencido y nos dice:

«No lo conozco. ¿Quién es?»

Introducimos su nombre (VELAZQUEZ), y a continuación nos pide que le digamos más cosas sobre él. Introducimos dos características más:

«Nació en Sevilla» y «Murió en Madrid»

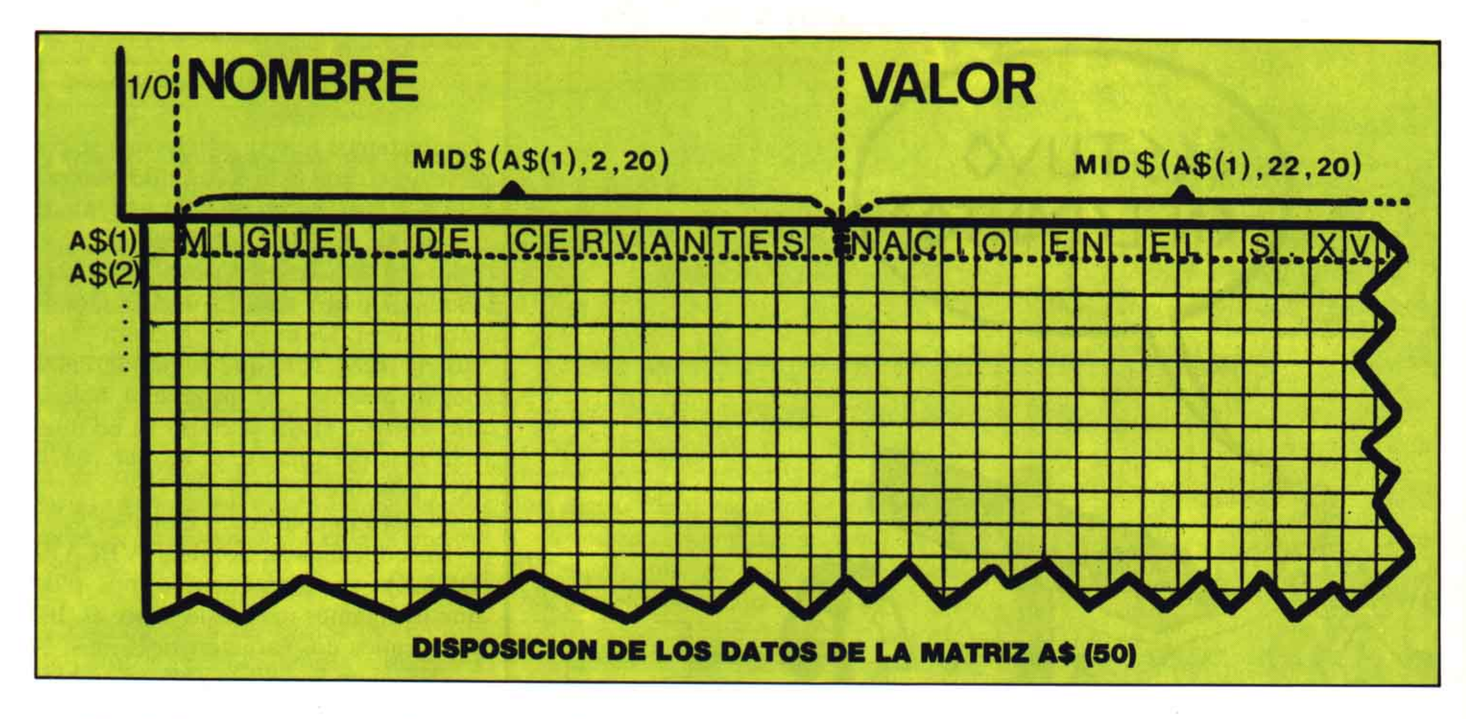
Finalmente la tabla de propiedad de **Velázquez** queda así:

VELAZQUEZ
fué pintor
trabajó en la corte
nació en Sevilla
murió en Madrid

Las características a las que habíamos contestado afirmativamente («fué pintor» y «trabajó en la corte») se habían almacenado previamente en el lugar que íba a ocupar en la matriz el nuevo personaje. Antes de hacer cada pregunta, el programa consultaba esta lista de propiedades nueva para no repetir una pregunta ya formulada. Si al final hubiera resultado que el personaje sí era conocido por la máquina (por ejemplo, si hubiéramos pensado en **Goya**), esta nueva lista se hubiera borrado, pues no se trataría de un personaje nuevo.

DESGLOSADO

Cuando ejecutéis el programa, observaréis que hace lo mínimo necesario para cumplir con su cometido, y que no pierde el tiempo en adornos ni virtuosismos. Esto es así porque lo hemos hecho eliminando todos los elementos superfluos (exceptuando, naturalmente, las sentencias REM), y organizando el listado todo lo que ha sido posible, para que no haya ningu-



DISPOSICION DE LOS DATOS DE LA MATRIZ A$ (50)

na dificultad en localizar, analizar y (éste es nuestro mayor deseo) modificar las rutinas.

Hasta ahora hemos utilizado la palabra «característica» para referirnos a los elementos que caracterizan a los personajes. Desde el punto de vista de nuestra semántica, esto es correcto, pero en el *argot* de la **Inteligencia Artificial** no lo es. A partir de ahora nos referiremos a ese concepto llamándole «valor».

Aquellos que no estén familiarizados con el tratamiento de matrices alfanuméricas encontrarán las explicaciones que vienen a continuación quizás demasiado complicadas. Si es así, recomendamos que se salten esa parte y practiquen un poco haciendo uso del manual. No es difícil de aprender.

La base de datos del programa se fundamenta en una matriz bidimensional de tipo alfanumérico (concretamente A$(50)), que hemos dividido imaginariamente de veinte en veinte caracteres por línea a fin de delimitar el espacio disponible para cada valor. Por esta razón, el programa no admite cadenas de más de veinte caracteres. Para hacerlo más gráfico, nos remitiremos a la figura 1. Cada línea de la matriz, A$(1) por ejemplo, consta de diez espacios imaginarios, más un espacio de un sólo caracter para indicar si el personaje en cuestión ha sido ya descartado o no (0/1). En el primer espacio se coloca el nombre, y en los nueve siguientes sus valores correspondientes, de forma que a cada personaje le corresponde una línea de la matriz.

Dicho esto, la primera sugerencia que se nos ocurre es ampliar la capacidad de la matriz para dar cabida a más personajes y/o más valores.

En cuanto al listado del programa, las sentencias REM ya indican claramente dónde empiezan y acaban las rutinas y cuál es su función. De todas formas, vamos a desglosarlas someramente.

La rutina 1000-1050 es la encargada de ordenar y repartir el trabajo a las demás. Procesa nuestras respuestas y envía el flujo del programa a la rutina 1100-1180 ó a la 1300-1360 según sea nuestra respuesta positiva o negativa respectivamente. Apréciese que ambas rutinas comparten la línea 1162 y siguientes (se ha hecho así para ahorrar memoria y acortar el listado).

El resto de las rutinas son las encargadas de introducir nuevos datos después de «darse por vencida» la máquina, salvar y cargar datos, pasar a un nuevo personaje, etc. Será misión del lector «desmenuzar» el programa línea por línea para llegar hasta el fondo de todas las rutinas e introducir las modificaciones que crea oportunas.

FUNCIONAMIENTO

Antes de que comencéis a teclear el programa, debemos advertir que sólo puede funcionar correctamente si el cursor está en modo para mayúsculas (Caps Lock), pues de lo contrario no reconocería la tecla pulsada en algunas sentencias condicionales. Naturalmente, esto podría remediarse fácilmente. Dejaremos que sea el lector quien lo haga.

Al comenzar, el programa almacena en memoria un sólo personaje (**Cervantes**), con unos cuantos valores:

- Nació en el siglo XVI
- Murió en el siglo XVII
- Escribió **El Quijote**
- Estuvo en Lepanto
- Era manco

Esto debemos tenerlo en cuenta cuando juguemos por primera vez. A medida que vayamos usando el programa, éste aprenderá muchos más personajes y sus correspondientes características (valores). Si no queremos perder los datos almacenados al apagar el ordenador, cuando el programa nos pregunte si queremos pasar a otro personaje, podremos acceder a las rutinas de carga y grabado pulsando «N». Si no podemos esperar a que el programa nos haga esa pregunta (la hace cuando hemos terminado con un personaje), podemos interrumpir el programa pulsando BREAK y hacer:

SAVE «CAS:SAB»

Cuando el programa nos haga una pregunta, basta con pulsar «S» para contestar afirmativamente, y «N» para

hacerlo negativamente. Si el programa no identifica el personaje que le proponemos, preguntará «No lo conozco. ¿Quién es?». En este caso teclearemos su nombre (no más de veinte caracteres), y después de que nos diga «Dime más sobre él», teclearemos sus características (hasta nueve) una por una, seguidas de ENTER . Cuando hayamos terminado con ellas, teclearemos FIN.

No nos cansaremos de decir que se trata de un programa abierto a vuestras manipulaciones. Si podéis introducir alguna mejora (seguro que sí), no dejéis de hacerlo.

En próximos números volveremos con otros interesantes programas **I.A.**, hasta entonces, esperamos que paséis muy buenos ratos con el **Sabelotodo**.

```
10 REM SABELOTODO
20 REM COPY E. DEL VALLE
   / INPUT
40 CLS: LOCATE 12,10
50 PRINT "SABELOTODO":
   LOCATE 12,12
60 PRINT "INPUT  MSX":
   LOCATE 1,22
70 PRINT "PULSA UNA TECLA"
80 K$=INKEY$
90 IF K$="" THEN GOTO 80
100 REM COMIENZO
120 CLS: PRINT "ESTOY LISTO"
130 PRINT "COMENZAR..........
    1": PRINT "SALVAR/CARGAR.
    ....2"
140 X=VAL (INKEY$)
150 IF X=0 THEN GOTO 140
160 ON X GOTO 9000,1750
1000 REM SELECTOR
1005 N$=MID$(A$(X),Y,20):IF
     N$="
     " AND R=1 THEN 1910:'N$=
     20 ESPACIOS
1006 IF N$="
        " AND R=0 THEN 1160
1007 K=INSTR(A$(L),N$)
1008 IF K<>0 THEN 1160
1010 CLS: PRINT "PERSONAJE ";
     L-1: LOCATE 0,21: PRINT
     N$;"?"
1020 K$=INKEY$
1030 IF K$="S" THEN 1105
1040 IF K$="" THEN 1020
1050 GOTO 1305
1100 REM LO ES
1105 R=1
1110 FOR N=X TO L-1
1120 IF INSTR(A$(N),N$)=0
     THEN MID$(A$(N),1,1)
     ="0"
1130 NEXT
1140 A$(L)=A$(L)+N$:
     LL=LL+20
1160 Y=Y+20: IF Y<183
     THEN 1005
1161 IF Y>183 AND R=1
     THEN 1910
1162 Y=22
1165 X=X+1:IF X>=L AND
     R=0 THEN 1800
1170 IF MID$(A$(X),1,1)="0"
     THEN 1165
1180 GOTO 1000
1300 REM NO
1305 R=0
1310 FOR N=X TO L-1
1320 IF INSTR(22,A$(N),N$)
     <>0 THEN MID$(A$(N),1,1)
     ="0"
1330 NEXT
1360 GOTO 1162
1400 REM LOAD
1410 LOAD "CAS:SAB"
1420 GOTO 1700
1500 REM OTRO
1530 LET T=0
1540 CLS: LOCATE 0,21: PRINT
     "DIME MAS SOBRE EL"
1550 LINE INPUT U$
1555 IF LEN (U$)>20
     THEN 1550
1558 IF LEN (U$)<1
     THEN 1550
1560 IF U$="FIN" THEN 1700
1570 A$(L)=A$(L)+U$+STRING$
     ((20-LEN(U$)),32):
     LL=LL+20
1575 IF LL>200 THEN 1700
1580 CLS: LOCATE 0,T:
     PRINT U$
1585 T=T+1: LOCATE 0,21
     :PRINT "DIME MAS
     SOBRE EL"
1590 GOTO 1550
1600 REM SAVE
1610 SAVE "CAS:SAB"
1620 GOTO 1700
1700 REM FIN
1702 L=L+1: LL=22: X=1: Y=22
1703 IF L>50 THEN CLS: PRINT
     "NO HAY MAS MEMORIA":
     STOP
1710 FOR N=1 TO L-1:
     MID$(A$(N),1,1)="1"
     :NEXT
1720 CLS: LOCATE 0,21:
     PRINT "SEGUIMOS CON
     OTRO ?"
1725 K$=INKEY$
1730 IF K$="" THEN 1725
1740 IF K$="N" THEN 1750
1745 GOTO 1000
1750 LOCATE 0,21: PRINT"PULSA
     1 PARA SALVAR PULSA
     2 PARA CARGAR"
1755 K$=INKEY$
1760 IF K$="1" THEN 1600
1770 IF K$="2" THEN 1400
1780 GOTO 1755
1800 REM NO LO SE
1810 CLS: LOCATE 0,20: PRINT
     "NO LO CONOZCO
     QUIEN ES ?":LINE INPUT
     U$
1820 IF LEN (U$)>20 THEN 1810
1830 IF LEN (U$)<1 THEN 1810
1835 U$=" "+U$
1840 A$(L)=U$+STRING$((21-LEN
     (U$)),32)+A$(L):
     GOTO 1500
1900 REM ES...
1910 CLS: LOCATE 0,20: PRINT
     "ES ";MID$(A$(X),2,20)
     ;"?"
1915 K$=INKEY$
1920 IF K$="N" THEN R=0:
     GOTO 1162
1930 IF K$="S" THEN
     GOTO 1950
1940 GOTO 1915
1950 A$(L)=""
1960 L=L-1: GOTO 1700
9000 REM DATOS
9010 CLEAR 10000: DIM A$(50)
9020 R=0:LL=22:L=2:X=1:Y=22
9030 READ B$: A$(1)=B$
9040 DATA 1MIGUEL DE CERVANTE
     S NACIO EN EL S.XVI   MU
     RIO EN EL S.XVII  ESCRIB
     IO EL QUIJOTE ESTUVO EN
     LEPANTO    ERA MANCO
     ESTUVO ENCARCELADO
9050 GOTO 1000
```

BUSCA TU NOMBRE

UN CICLOMOTOR VESPINO

* ANTONIO FERNANDEZ DELGADO, ALCALA DE HENARES (MADRID).

UNA CADENA HIFI DE SPECTRAVIDEO

* JUAN LOPEZ SALA, ALICANTE.
* EMILIO DE ANDRES DE LA VEGA, MADRID.
* J. LORENZO LABO SOLANA, R. DE CAMARGO (CANTABRIA).

UN LP DE GABINETE CALIGARI

* SANTIAGO CAMPAYO CUESTA, BADALONA (BARCELONA).
* ALBERTO VILARMAU ESPINAL (BARCELONA).
* LUIS ALBERTO CARRION MORENO, VALLADOLID.
* MANUEL CASTILLEJOS VAYA, MADRID.
* MANUEL ALONSO MARTINEZ, BILBAO.
* PEDRO BARAHONA CORDERO, COSLADA (MADRID).
* MIGUEL SUNE RODRIGUEZ. BARCELONA.
* JOSECHU FERNANDEZ BAREA, SANTURCE (VIZCAYA).
* JOSE LUIS MARTINEZ MUNOZ, STA. COLOMA DE GRAMANET (BARCELONA).
* GONZALO NOVOA SOTELO, ORENSE.

UN LOTE DE SOFTWARE DE ERBE

* SANTIAGO ESTEFANIA BARRIO, GETXO (VIZCAYA).
* J. MANUEL COLOMINA MESTRES, STA. COLOMA DE GRAMANET (BARCELONA).
* VICTORINO RODRIGUEZ VALDESOGO, LEON.
* J. MANUEL PEREIRA GARCIA, BARCELONA.
* MA. ANGUSTIAS DELGADO, GRANADA.
* MANUEL TERRER MENA, CARTAGENA (MURCIA).
* MIGUEL SALAS GARCIA, MADRID.
* J. CARLOS DE LA LLANA SANTO, REUS (TARRAGONA).
* SERGIO MARTINEZ COLLDEFORN, BARCELONA.
* ROBERTO TORRIJOS SANTOS, TALAVERA (TOLEDO).

TRES PROGRAMAS DE COMPULOGICAL

* FERNANDO GONZALEZ RODRIGUEZ, LA OROTAVA (TENERIFE).
* J. ANTONIO POZO GALLETERO, CHIVA (VALENCIA).
* A. MANUEL YANEZ VALEIRO, BILBAO.
* JORDI MARZO VIDAL, BARCELONA.
* JOAN CARRASCO DOMINGUEZ, TERRASA (BARCELONA).
* ANSELMO FERNANDEZ BERODAS, MALAGA.
* FRANCISCO MACIA LACUSA, BARCELONA.
* ANDRES ORTIZ MENDEZ, MARTOREL (BARCELONA).
* ESTEBAN HORCAJO ARROYO, S.S. DE LOS REYES (MADRID).
* JAVIER SUBIRATS CERVERA, REUS (TARRAGONA).

TRES PROGRAMAS DE PROEINSA

* JORDI PRUNES CASTERAS, TERRASA (BARCELONA).
* DAVID BRAVO CALVO, BARCELONA.
* ANTONIO GRIMALT ANDREU, MANACOR (MALLORCA).
* RUBEN DOMINGUEZ LOPEZ, AGUADULE (ALMERIA).
* SALVADOR DE LA FUENTE FERNANDEZ, VALENCIA.
* ALFONSO MANTECON LAGE, LAS ROZAS (MADRID).
* MIGUEL LA TORRE ROVIROSA, VILAFRANCA DEL PENEDES (BARCELONA).
* M. ANGEL FLORES ACUNA, LA CORUNA.
* F. CARLOS GAITAN ALFARO, CORDOBA.
* J. MIGUEL TORES CARO, CADIZ.

TRES PROGRAMAS DE MICROUNO

* MANUEL PINEIRO BESADA, PORRINO (PONTEVEDRA).
* A. JOSE PALACIOS SOLER, VALENCIA.
* SEBASTIAN MONZON PEREZ, SEVILLA.
* J. CARLOS CAPAFONS CUBERO, SITGES (BARCELONA).
* JAVIER HINOJOSA RUIZ, CASTELLAR DEL VALLES (BARCELONA).
* ALFREDO CANTELI SUAREZ, OVIEDO (ASTURIAS).
* R. ALBERTO GARGALLO CASCAN, BARCELONA.
* SANTIAGO FERREIRO PARAMO, LA CORUNA.
* ANTONIO GALAN ELEZ, MADRID.
* JESUS VALERO ALCARAZ, MADRID.

CUATRO PROGRAMAS DE SERMA

* ANTONIO GOMEZ AGUILAR, MALAGA.
* PAQUI RODRIGUEZ GONZALEZ, BADAJOZ.
* FERNANDO ROJAS PALACIOS, ALGECIRAS (CADIZ).
* RAFAEL ABENJO PLAZA, MARACENA (GRANADA).
* J. MANUEL PARRA BARBERO, LA HERRADURA (GRANADA).
* PABLO HERRERA PEREZ GRUESO, MORA (TOLEDO).
* TEOFILO DEL RINCON GONZALEZ, PUIG DEN VALS (IBIZA).
* JOSE LUIS GONZALEZ OLEA, FUENLABRADA (MADRID).
* JAIME SOLA FLOX, VILANOVA I LA GELTRUS (BARCELONA).
* ALVARO DAPENA URCOLA, ZARAGOZA.

UN KIT ROBOTICO DE FERRE-MORET

* JOSE JULIO VELAZQUEZ, LEJONA (VIZCAYA).

UN JOYSTICK DE COMPULOGICAL

* MIGUEL MILLAN DE LA TORRE, S.JUAN DESPI (BARCELONA).
* VICENTE MORALEDA HIDALGO, SEVILLA.
* RAMON JUNQUERA SOBRON, GALDAKANO (VIZCAYA).

UN JOYSTICK DE MICROUNO

* JOSE C. RODRIGUEZ ESCOBAR, MADRID.
* BENITO SANTAMARIA ROMERO, MADRID.
* JOSE ANTONIO GONZALEZ CUETO, GIJON (ASTURIAS).
* DANIEL ROSS ESTRAGUEZ, BARCELONA.
* ANTONIO DIAZ GONZALEZ, SANTANDER.
* EDUARDO TORRECILLAS VALVERDE, BADALONA (BARCELONA).
* PEDRO LANCILLA TELLA, ZARAGOZA.
* V. ANTONIO UCEDA ALVAREZ, GETAFE (MADRID).
* J. MANUEL VIGO COUSINO, VIGO (PONTEVEDRA).
* MARIANO LOPEZ LOPEZ, COLMENAR VIEJO (MADRID).
* JAVIER CORONA DELGADO, PAMPLONA.
* FCO. JAVIER DIAZ GARZON, BARCELONA.
* J. MANUEL TRIAS COMPANY, SOLLER (MALLORCA).
* PABLO VAQUERO SEPULVEDA, AVILA.
* JESUS OCANA GOMEZ, MADRID.

UN CRONOMETRO DE MICROUNO

* JUAN PENA SANCHEZ, ISLA CRISTINA (HUELVA).
* ANGEL GOMEZ LOPEZ, LUGO.
* IGNACIO TIERNO MARTINEZ, BARCELONA.
* MA. LUISA SANZ AGUADO, LLAVERNERS (BARCELONA).
* J. LUIS BLASCO CABAL, MADRID.

UN LIBRO DE ANAYA MULTIMEDIA

* ANTONIO CABEZA HERNANDEZ, TALAVERA (TOLEDO).
* JAVIER MARTINEZ CORMENEZANA, BILBAO.
* MIGUEL SERRANO DEL MAZO, CACERES.
* ECKART SCHILDER, BREXO (LA CORUNA).
* SANTIAGO DIAZ PEREZ, PORTUGALETE (VIZCAYA).
* RICARDO SEMPERE PALAU, QUART DE POBLET (VALENCIA).
* FRANCISCO TINEO RAMIREZ, MATARO (BARCELONA).
* MIGUEL ANGEL FERNANDEZ, MADRID.
* JORGE ABAD GIL, MADRID.
* EVA GOMEZ RODRIGO, SALAMANCA.
* F. JAVIER VAZQUEZ SERRANO, CORDOBA.
* J. CARLOS GARRIDO BERNARDEZ, ORENSE.
* J. MIGUEL MELLADO BREY, SAN ROQUE (CADIZ).
* SANTIAGO HERNANDEZ SANCHEZ, FUENLABRADA (MADRID).
* GINES CLIMENT PEREZ, VILLAJOYOSA (ALICANTE).

liminar, más otros que te permitan enlazar el juego.

Al dibujar este mapa, acuérdate de marcar los sentidos en que se puede circular en cada habitación, porque puede ser que haya puertas que quieras que sólo puedan cruzarse en una dirección, acompañadas de un mensaje tal como éste:

LA PUERTA SE CIERRA DE GOLPE JUSTO DETRAS DE TI

Las líneas de trazos que hay junto al cuarto oscuro indican que el aventurero sólo podrá ir en esa dirección si se cumplen ciertas condiciones. En este caso, la condición es que el aventurero tenga la lámpara y la haya encendido para poder ver las salidas.

Es muy difícil predecir cuántos lugares se pueden meter en una cantidad dada de RAM. La dificultad surge porque hay muchas cosas que se disputan el espacio de memoria en el programa de la aventura: descripciones de lugares, palabras que quieres que el programa reconozca, número de objetos y lo que quieres que se haga con ellos, número de enigmas y su complejidad, etc.

Cuando hayas escrito unas cuantas aventuras pequeñas y comprobado cuánta memoria ocupan, tendrás una idea de lo que puedes meter en tu máquina.

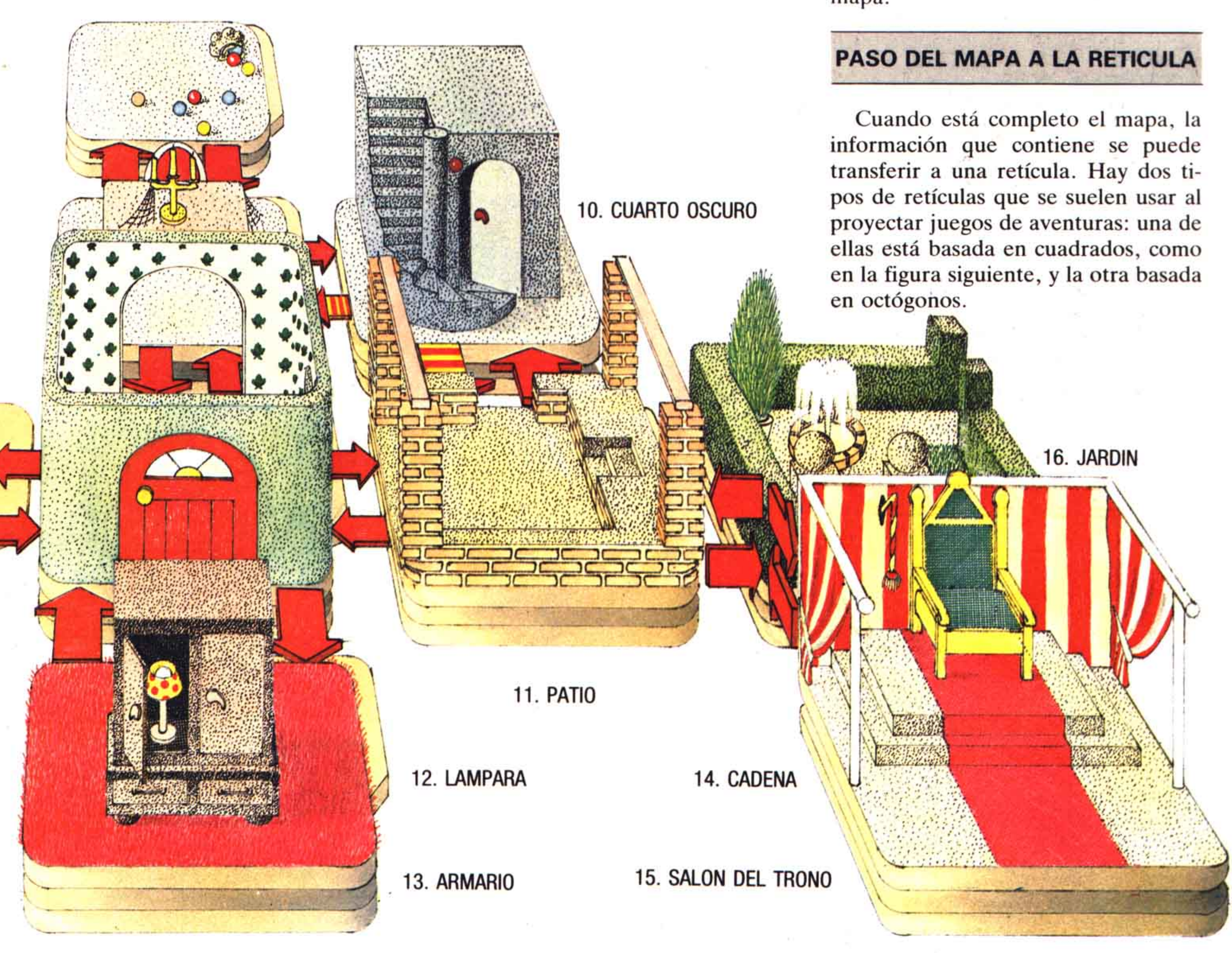
Los poseedores de un ordenador de 16K pronto descubrirán que es casi imposible escribir una aventura en gran escala en una cantidad tan pequeña de RAM. Sin embargo, la aventura que desarrollaremos en los siguientes capítulos sólo tiene unos pocos lugares —12 en total— por lo que es suficientemente pequeña para no causarte dolores de cabeza.

El mapa para la búsqueda de la joya del globo ocular podría ser algo parecido al de la figura. En esta fase las descripciones de los lugares han de mantenerse cortas. Se han dibujado los enlaces y se ha decidido el punto de partida. Esto es muy importante, ya que afecta a la manera en que se aborda la aventura, al orden en que aparecen los objetos y se plantean los *puzzles*.

También están marcados los objetos en sus lugares. Los objetos que aparecerán más tarde, por ejemplo la joya, es mejor listarlos a un lado del mapa.

PASO DEL MAPA A LA RETICULA

Cuando está completo el mapa, la información que contiene se puede transferir a una retícula. Hay dos tipos de retículas que se suelen usar al proyectar juegos de aventuras: una de ellas está basada en cuadrados, como en la figura siguiente, y la otra basada en octógonos.



En las aventuras más sencillas, sólo se podrá salir de cada lugar en una de estas cuatro direciones: Norte, Sur, Este y Oeste (así ocurre en la aventura del globo ocular). Si tu aventura es de este tipo, tienes que transferir a la rejilla cuadrada toda la información para que resulte equivalente al mapa. Más adelante desarrollaremos esto con detalle. Si has utilizado salidas que incluyen Nordeste, Noroeste, Sureste y Suroeste, tienes que emplear la retícula octogonal, aunque este tipo de rejillas resulta complicado.

La última variante de la rejilla se produce si has decidido también subir y bajar. En este caso la mejor solución es utilizar una rejilla separada para cada «nivel» de aventura.

La aventura del ojo precioso está basada sobre una retícula cuadrada, permitiendo sólo salidas por el Norte, Sur, Este y Oeste. A menos que haya una necesidad real de otras direcciones, este tipo de aventuras resulta muy adecuado y hay una manera de obviar las subidas y bajadas. Si utilizas una descripción con frases como

HAY UNA ESCALERA QUE BAJA HACIA EL OESTE

puedes usar la respuesta normal del Oeste o la rutina adecuada.

LA RETICULA

Esta aventura requiere una retícula cuadrada de 6×4 como puedes comprobar contando el número máximo de lugares de tu mapa, de arriba a abajo y de derecha a izquierda. Antes de empezar a trasladar todos los detalles a la retícula, asegúrate de que has numerado todos los cuadrados. Empieza con el número 1 en la parte superior izquierda, y prosigue hasta llegar a la parte inferior derecha. Cuando hayas numerado todos los cuadrados y transferido todos los detalles, la retícula será como la de la figura.

EMPIEZA EL PROGRAMA

Ahora tienes una línea histórica y una retícula completa, puedes empezar con el programa.

El primer paso es teclear las descripciones de los lugares, basándote en tu retícula. Tienes que decidir cómo van a ser de largas. Intenta sugerir lo más posible el ambiente de la aventura sin derrochar memoria. Aparte de las descripciones de los lugares, hay que decirle al ordenador en qué direcciones se encuentran las salidas.

Aquí tienes al fin las primeras secciones del programa. Los números de líneas tan altos te permitirán disponer de suficiente espacio en las anteriores secciones de programa a medida que vayas desarrollando el juego.

Teclea esta sección y almacénala (SAVE) en cinta:

```
5000 REM ** DESCRIPCION DE
     UBICACIONES
5010 REM ** UBICACION 4
5020 PRINT"ESTAS EN EL
     EXTERIOR DE UN GRAN
     EDIFICIO"
5030 N=0:E=0:S=1:0=0:RETURN
5040 REM ** UBICACION 7
5050 PRINT"ESTAS EN UN RIO
     TURBULENTO"
5060 N=0:E=1:S=0:0=0:RETURN
5070 REM ** UBICACION 8
5080 PRINT"ESTAS EN UN BOSQUE
     PETRIFICADO"
5090 N=0:E=0:S=1:0=1:RETURN
5100 REM ** UBICACION 10
5110 PRINT"ESTAS EN UNA SALA
     POLVORIENTA"
5120 N=1:E=1:S=1:0=0:RETURN
5130 REM ** UBICACION 11
5140 PRINT"ESTAS EN UN CUARTO
     OSCURO"
5150 IF OB(6)<>-1 OR LA<>1
     THEN N=0:E=0·S40:0=0
5155 IF OB(6)<>-1 OR LA<>1
     THEN PRINT"NO SE VEN LAS
     SALIDAS EN LA OSCURIDAD"
     :RETURN
5160 N=0:E=0:S=1:0=1:RETURN
5170 REM ** UBICACION 14
5180 PRINT"ESTAS EN UN CAMINO
     LLENO DE BARRO"
5190 N=1:E=1:S=0:0=0:RETURN
5200 REM ** UBICACION 15
5210 PRINT"ESTAS EN LA PUERTA
     DE LA CIUDAD OCULTA"
5220 N=0:E=1:S=0:0=1:RETURN
5230 REM ** UBICACION 16
5240 PRINT"ESTAS EN EL HALL
     DE ENTRADA"
5250 N=1:E=1:S=1:0=1:RETURN
5260 REM ** UBICACION 17
5270 PRINT"ESTAS EN EL
     PATIO"
5280 N=1:E=1:S=0:0=1:RETURN
5290 REM ** UBICACION 18
5300 PRINT"ESTAS EN EL
     JARDIN"
5310 N=0:E=0:S=1:0=1:RETURN
5320 REM ** UBICACION 22
5330 PRINT"ESTAS EN EL
     ARMARIO"
5340 N=1:E=0:S=0:0=0:RETURN
5350 REM ** UBICACION 24
5360 PRINT"ESTAS EN LA SALA
     DEL TRONO"
5370 N=1:E=0:S=0:0=0:
     RETURN
```

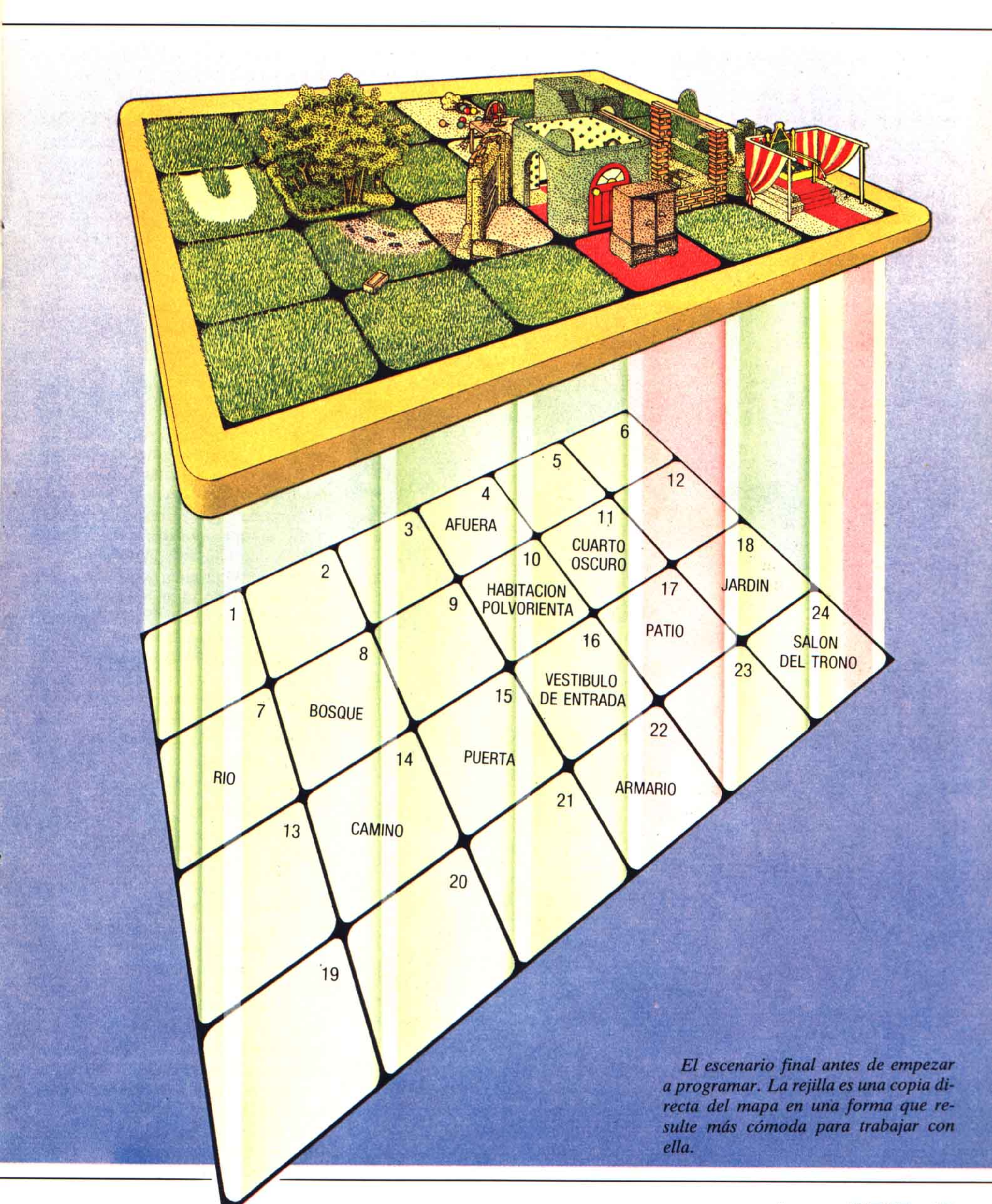
No te preocupes por el abundante uso de las sentencias REM y toda la memoria que consumen. En esta primera etapa del desarrollo del programa es muy importante saber qué es lo que hace cada parte del mismo, o a qué número de lugar se refiere una descripción particular. Siempre puedes quitarlas más adelante.

Después de cada línea con la descripción de un lugar, hay otra línea que contiene información sobre sus posibles salidas. Las variables N, S, E y O significan Norte, Sur, Este y Oeste. Estas variables pueden tomar uno de dos valores posibles: 0 significa que no hay salida en esa dirección, mientras que 1 significa que hay salida.

Finalmente, después de cada sección de este programa hay un RETURN, debido a que cada descripción de lugar se llama mediante una sentencia GOSUB.

La sentencia IF...THEN que hay en la sección del cuarto oscuro es para comprobar si el aventurero tiene la lámpara, pero la descripción de las variables se comentará más adelante cuando nos ocupemos de los objetos.

En el próximo capítulo veremos la manera en que se mueve el aventurero por los distintos lugares.

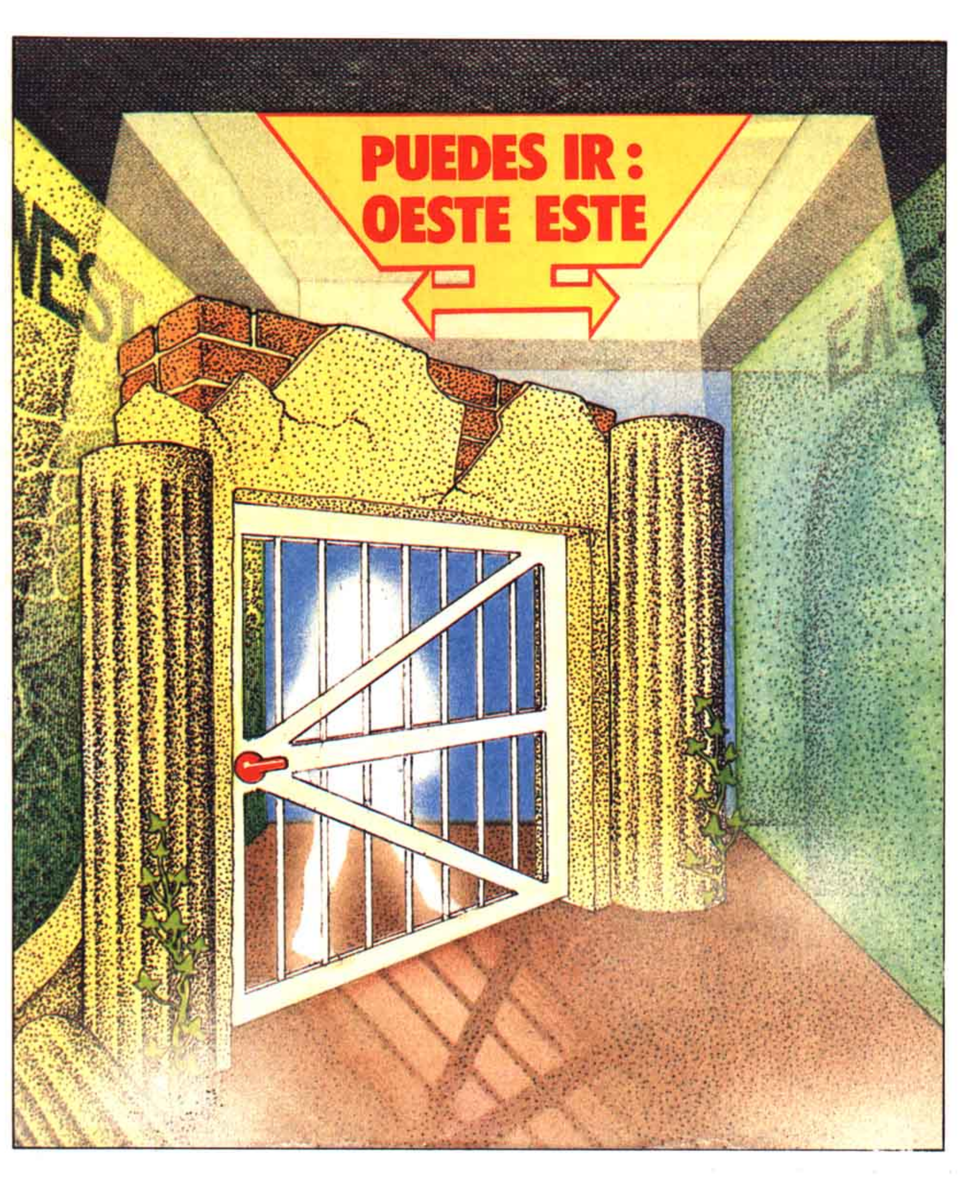


El escenario final antes de empezar a programar. La rejilla es una copia directa del mapa en una forma que resulte más cómoda para trabajar con ella.

UNA AVENTURA MOVIDA

- PRESENTACION DE DIRECCIONES
- TRATAMIENTO DE LAS INSTRUCCIONES
- MOVIMIENTO A TRAVES DE LA AVENTURA
- BLOQUEO DE LOS MOVIMIENTOS IMPOSIBLES

Una parte de la diversión de jugar juegos de aventuras es la posiblidad de explorar mundos nuevos y extraños sin salir de casa. Veamos la manera de iniciar estas exploraciones.



Ahora que ya has tecleado un programa que contiene todas las descripciones de los ambientes, es el momento de hacer que el aventurero pueda explorarlos, desplazándose de un lugar a otro. Tienes que prever todos los movimientos posibles en cada punto, y fijar unos límites, basándote en el grado de dificultad y en las pistas que se van recogiendo a medida que progresa la aventura.

En esta ocasión, para ampliar el desarrollo de tu programa de aventuras, presentaremos rutinas con la descripción correcta de los ambientes, junto con las posibles salidas a los mismos. Al jugador se le pedirá que introduzca respuestas, y verás la manera de escribir una seccion de programa para que el jugador se vaya moviendo por el interior del fantástico mundo, en base a las elecciones hechas, a medida que progresa la aventura.

EL PUNTO DE PARTIDA

Lo primero que el ordenador tiene que saber es dónde está el aventurero en cada momento del juego. Para hacer esto el programa mira al valor que va tomando una variable llamada L (de lugar). A esta variable se le asigna un valor que corresponde al ambiente en que se encuentra el aventurero después de cada movimiento.

Para empezar pues la aventura, tienes que decirle al ordenador dónde quieres que se encuentre el aventurero en el momento de empezar.

Aquí tienes la primera parte de un programa que se encarga de esto. Carga (con LOAD) la sección que tecleaste la última vez, y añádele las siguientes líneas:

```
270 REM ** POSICION DE
    COMIENZO
280 L=15
290 GOTO 330
```

El número 15 corresponde al lugar en que se encuentra la puerta de la ciudad escondida. Si quieres que la aventura empiece en otro lugar, basta con que modifiques el valor de L. Enseguida veremos la forma de modificar el valor de L durante el juego, según el lugar atravesado. Pero antes de que el aventurero pueda empezar a moverse, hay que decirle al ordenador hacia dónde tiene que ir. Para ello el aventurero utilizará el teclado, a través del que tendrá que introducir una o varias palabras.

RESPUESTAS

Para que el ordenador pueda entender adecuadamente tus respuestas, y actuar en consecuencia, tienes que proporcionarle una lista de todas las palabras que puede reconocer.

En esta etapa del desarrollo, basta con que reconozca las cuatro direcciones NORTE, SUR, ESTE y OESTE. Podemos meterlas en una matriz R$, que contiene los datos de cada dirección de respuesta.

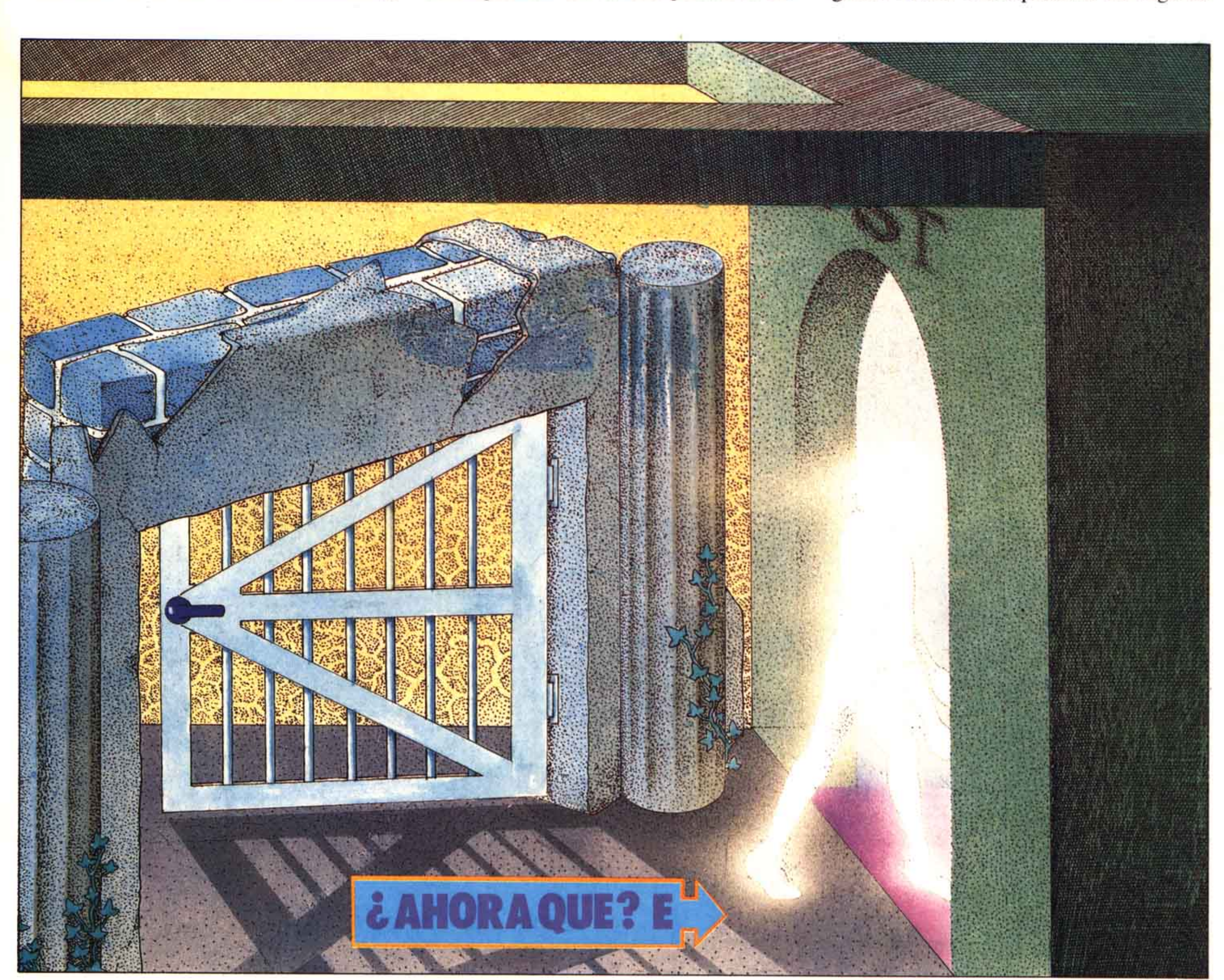
```
110 REM ** MATRICES DE
    RESPUESTAS
120 DIM R$(19),R(19):KEY OFF:
    WIDTH 37:CLS
130 FOR K=1 TO 4:READ R$(K)
    ,R(K):NEXT
150 DATA NORTE,1,SUR,1,ESTE,1
    ,OESTE,1
```

En la línea 120 se dimensionan las matrices, de modo que contengan todas las respuestas necesarias para el juego, dado que por el momento sólo necesitas utilizar las direcciones, sólo se emplearán los cuatro primeros elementos de las matrices R$ y R. El bucle FOR ... NEXT de la línea 130, va desde uno hasta cuatro, leyendo tanto en R$ como en R. Las direcciones y sus números están en las sentencias DATA de la línea 150.

Pero evidentemente esta información no le sirve para nada al jugador, a menos que el ordenador le diga previamente dónde se encuentra.

LA BUSQUEDA DE UN LUGAR

Para que el aventurero pueda saber adonde ha ido a parar después de cada uno de sus movimientos, hay que suministrarle una descripción de los lugares. Estas descripciones de lugares ya las tienes tecleadas; sólo te hace falta una rutina para extraer la descripción que corresponda al valor de la variable L, el número de lugar. Aquí es donde resultan útiles las líneas REM que introdujiste anteriormente.



```
300 REM ** ENCONTRAR
    UBICACION
310 CLS
330 IF L<11 THEN ON L GOSUB 0
    ,0,0,5020,0,0,5050,5080,0
    ,5110:
```




```
    GOTO 400
340 IF L<21 THEN ON L-10
    GOSUB 5140,0,0,5180,5210,
    5240,5270,5300,0,0
    :GOTO 400
350 IF L<26 THEN ON L-20
    GOSUB 0,5330,0,5360
```

Antes de escribir este tipo de rutinas, tienes que asegurarte del número correspondiente a cada descripción de lugar. A partir del lugar número 1, tienes que ir haciendo una lista de los números de líneas de cada descripción. Si para alguno de los ambientes no hay ninguna descripción especial, asígnale un 0. En nuestra aventura, por ejemplo, no hay descripciones para los lugares 1, 2 y 3, pero sí la hay para el lugar 4.

Ahora que tienes la lista de los números de líneas, puedes empezar a escribir la rutina. Es una sencilla secuencia de operaciones de control del valor de L por medio de una sentencia ON ... GOSUB.

La lista de números de líneas va desde la línea 330 a la 350, comenzando por el lugar 1 al principio de la línea 330 y terminando con el lugar 24 al final de la línea 350.

PRESENTACION DE LAS DIRECCIONES

Aparte de conocer las descripciones de los lugares, el jugador de la aventura querrá saber qué salidas tiene. Como en cada lugar no le será posible moverse en todas direcciones, el programa debe tener alguna forma de controlar las informaciones asociadas

con la dirección, es decir las variables N, S, E y O. La siguiente rutina dirá al aventurero qué direcciones son posibles:

```
390 REM ** MUESTRA
DIRECCION
400 IF L<>11 OR (LA=1 AND
    OB(6)=-1) THEN PRINT
    :PRINT"PUEDES IR ";:GOTO
    410
405 GOTO 460
410 IF N>0 THEN PRINT TAB(11)
    "NORTE"
420 IF S>0 THEN PRINT TAB(11)
    "SUR"
430 IF E>0 THEN PRINT TAB(11)
    "ESTE"
440 IF O>0 THEN PRINT TAB(11)
    "OESTE"
```

La rutina se limita a comprobar el valor de las variables N, S, E y O, basándose en el mapa de direcciones. Si el valor de las variables es mayor que cero, la dirección es posible y se presenta la correspondiente salida.

Esta rutina puede ser incorporada tal como está en cualquier aventura basada en una retícula de cuadrados.

INSTRUCCIONES

Ahora que el aventurero ya conoce las direcciones que puede tomar, hay que hacerle algunas sugerencias. La siguiente sección de programa le preguntará al jugador ¿QUE VAS A HACER AHORA?

```
450 REM ** INSTRUCCIONES
460 PRINT:INPUT"AHORA
QUE";I$
470 GOSUB 3010
```



Se trata de una rutina de entrada muy sencilla. A la respuesta suminis-

ESTAS EN LA ENTRADA

trada la llamamos I$. El ordenador debe comprobar que la respuesta es correcta y actuar en consecuencia. La línea 470 envía el control a la subrutina de la línea 3010, que es la encargada de procesar la respuesta dada por el jugador.

```
3000 REM ** COMPRUEBA
     INSTRUCCION
3010 N$="":FOR Z=1 TO LEN(I$)
     :IF MID$(I$,Z,1)=" "
     THEN I=Z:GOTO 3020
3015 NEXT:I=0
3020 IF I=0 THEN V$=I$
     :GOTO 3050
3030 V$=LEFT$(I$,I-1)
3040 N$=MID$(I$,I+1)
3050 I=0
3060 FOR K=1 TO 19
3070 IF V$=LEFT$(R$(K),
     LEN(V$)) THEN I=R(K)
     :I$=LEFT$(V$,1)
3080 NEXT
3090 RETURN
```

Esta rutina comprueba si I$ contiene dos palabras. De ser así, llama V$ a la primera y N$ a la segunda. Hemos puesto V$ por verbo, como COGER, MATAR, o LLEVAR, y también sirve para las direcciones como NORTE, SUR, ESTE Y OESTE. N$ significa nombre, y vale para designar los objetos que aparecen en la aventura. El programa utiliza la sentencia MID$ (líneas 3010 a 3015) para examinar si hay un espacio entre V$ y N$.

Si se encuentra un espacio, la línea 3020 designa a las dos partes de I$ con las etiquetas N$ y V$. Si no se encuentra espacio, la propia línea 3020 llama V$ a toda I$.

El resto de la subrutina (líneas 3060 a 3080) compara las respuestas obtenidas con las de la matriz R$, que contiene las direcciones de las respuestas. Más adelante veremos cómo puede ampliarse para contener una serie adicional de verbos. Si en la línea 3070 se detecta una coincidencia, se pone en I el correspondiente valor de R. En fases posteriores del programa la máquina puede saber si ha habido una coincidencia examinando si I es mayor que cero. La última parte de la línea 3070, toma la primera letra de V$ y la llama I$, lo que servirá después para el movimiento del aventurero.

Esta subrutina puede utilizarse casi sin limitaciones en cualquier juego de aventuras. Probablemente la única modificación que le hagas, estará relacionada con la longitud del bucle FOR ... NEXT de la línea 3060.

P y R

¿Entenderá el programa respuestas de dirección tales como NORTE o IR NORTE, además de letras solas, tal como N?

La rutina de comprobación de instrucciones, situada en las líneas 3.000 a 3.090, está escrita de forma que admite palabras de cualquier número de letras. Cuando dos palabras van separadas por un espacio el programa las maneja de forma independiente.

La línea 3.070 toma, como respuesta de dirección, la primera letra. Siempre que esta primera letra sea N, S. E u O (en mayúscula), el programa entenderá lo que quiere decir el jugador, independientemente de cual sea la respuesta completa. Así pues: N, No y NORTE son respuestas válidas, pero no lo es IR NORTE.

Sin embargo, no hay nada que te impida hacer adiciones que permitan al aventurero utilizar respuestas del tipo IR NORTE. Más adelante veremos cómo maneja el programa los verbos y los nombres.

Lo que habrá que hacer será añadir IR a la lista de verbos y escribir una rutina de procesamiento de esta nueva forma verbal.

MOVIMIENTO

Para que el aventurero pueda explorar todos los lugares, basta que añadas al programa una rutina que transforme la variable de lugar L, según el contenido de I$. Aquí la tienes:

```
1000 REM ** RUTINA DE
     MOVIMIENTO
1010 IF I$="N" AND N>0 THEN
     L=L-6:GOTO 310
1020 IF I$="E" AND E>0 THEN
     L=L+1:GOTO 310
1030 IF I$="S" AND S>0 THEN
     L=L+6:GOTO 310
1040 IF I$="O" AND O>0 THEN
     L=L-1:GOTO 310
1050 REM ** SI NO HAY
     UBICACION POSIBLE EN TAL
     DIRECCION
1060 PRINT:PRINT"LO SIENTO"
     :PRINT"NO PUEDES IR EN
     ESA DIRECCION!"
     :GOTO 330
```

Recuerda que la aventura estaba basada en una retícula de seis por cuatro. El movimiento implica un cambio del valor de L, que depende del tamaño de la retícula. Moverse hacia el Norte o hacia el Sur equivale a sumar o restar 6 al valor de L para desplazarse una línea hacia arriba o hacia abajo en la retícula. Análogamente, el movimiento hacia el Este o hacia el Oeste es equivalente a sumarle o restarle 1 al valor de L.

Las líneas 1010 a 1040 comprueban las direcciones en I$ y ajustan L en consecuencia. Sólo es posible el movimiento si hay una salida que coincide con I$. Las salidas fueron definidas en las líneas inmediatamente siguientes a las descripciones de los lugares.

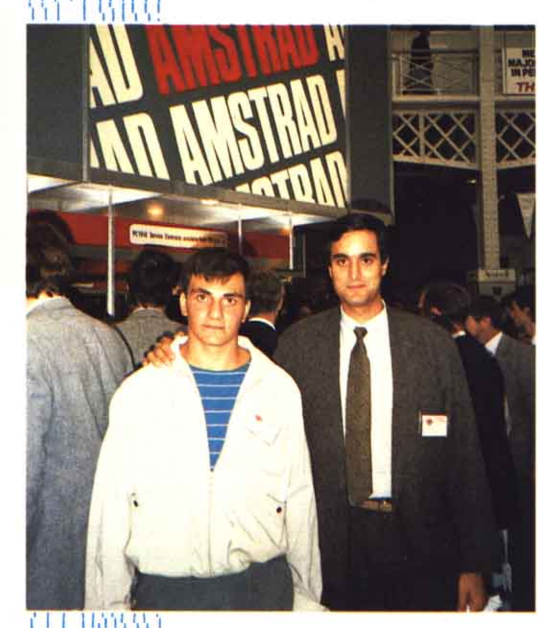
Si no existe salida en la dirección en que el aventurero quiere salir, la línea 60 presenta el mensaje ¡LO SIENTO! ¡NO PUEDES SALIR POR AHI!

Para utilizar esta rutina con una aventura diferente, el único cambio que tendrás que hacer tendrá que ver con el tamaño de la retícula. En tal caso, habrá que modificar las líneas 1010 a 1030, con arreglo a las líneas de la retícula.

Almacena ahora el programa (con SAVE) dejándolo listo para ser utilizado en el próximo capítulo.

DESCUBRE AL ASESINO

Muchos abandonaron en el camino, pero otros (pocos) llegasteis hasta el final en el reto planteado, Finalmente hubo que realizar un sorteo, imparcial por supuesto, al que asistimos elementos de la redacción de INPUT y personal de Anaya Multimedia.
Los mensajes planteados tenían como respuesta lo siguiente:



"ES UNA VIOLENCIA FINGIDA PUES SE TRATA DE UN TIPO DE CAJA INACCESIBLE A CUALQUIER TIPO DE PALANCA".

" LAS ATADURAS NO SON DE UN PROFESIONAL DA LA IMPRESION DE QUE TENIAN MIEDO DE LESIONARLE".

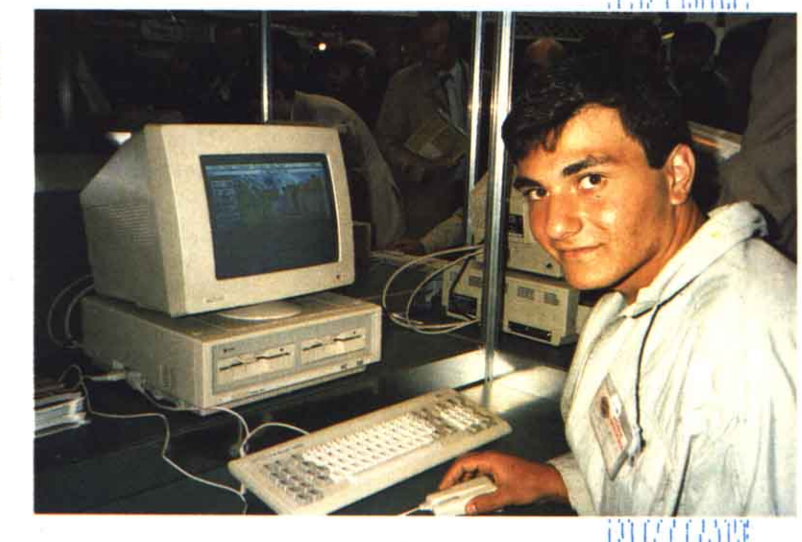
"TIENE LA COSTUMBRE DE MORDER LAS COLILLAS DE LOS CIGARROS QUE FUMA SIN PARAR HASTA DESHACERLAS".

"EN EL REGISTRO EN LA CASA DE LA Victima aparecieron las joyas habilmente CAMUFLADAS BAJO EL SUELO DEL DORMITORIO".

"A PESAR DE ESTAR BASTANTE QUEMADA EN LA COLILLA SE PODIAN DISTINGUIR PERFECTAMENTE LAS MARCAS DEJADAS POR LOS INCISIVOS DE UN FUMADOR NERVIOSO".

El intrépido y sagaz descifrador de claves se llama Manuel Bautista López. Todo un hallazgo para los servicios de inteligencia de cualquier pais, que no dudamos se lo rifarán cuando este ejemplar caiga en sus manos. Todo un criptofuturo por delante.

En la fotografía podemos apreciar a nuestro afortunado Manuel disfutando de su estancia en el PCW Show de Londres. Le acompana Santos Rodriguez, director de producto y delegado de Anaya Multimedia para el significativo evento.



ORDENADORES, PUZZLES Y MATEMATICAS

- TIPOS DE PUZZLES
- USO DEL ORDENADOR
- ECUACIONES SIMULTANEAS
- PRUEBA Y ERROR
- TRUCOS MATEMATICOS

Agudiza tu ingenio y pon a prueba tu destreza en programación con estos difíciles rompecabezas. Al mismo tiempo aprenderás unas cuantas técnicas nuevas para resolver sistemas de ecuaciones.

A estas alturas de nuestra andadura en INPUT, ya debes tener una buena base en unas cuantas técnicas de programación en BASIC. Dado un problema cualquiera, deberías ser capaz de escribir un programa para resolverlo (en el supuesto de que realmente pueda ser resuelto). Pero a menos que tengas algún *hobby* especial o algún proyecto entre manos que haga uso del ordenador, con frecuencia es difícil pensar cosas con las que ponerse a hacer pruebas.

Los rompecabezas ofrecen un desafío que suele ser bienvenido, ya que constituyen una forma entretenida de practicar la programación y de mantener al día tus habilidades. Han sido muy populares durante millares de años. Se sabe que los egipcios disfrutaban resolviendo rompecabezas y que los antiguos griegos eran extremadamente aficionados a los rompecabezas y paradojas matemáticas y lógicas. De hecho muchos problemas que empezaron como un simple divertimento condujeron posteriormente a importantes descubrimientos en matemáticas y en otras ciencias.

La resolución de un problema con éxito te hará sentir una sensación de satisfacción y, si participas en alguno de los cada vez más abundantes concursos organizados por las revistas de informática, puedes ganar muchos premios. Además, la solución de estos rompecabezas te enseñará sobre programación mucho más de lo que puedes aprender por simples lecturas sobre el tema ya que te obliga a trabajar con tus propios métodos y a poner a prueba tus propias ideas.

PUZZLES PARA ORDENADOR

Existen muchos tipos de rompecabezas, pero no todos ellos resultan adecuados para ser resueltos por medio de ordenador. Muchos requieren una idea feliz o un poco de razonamiento lógico. Un ejemplo clásico del tipo que requiere razonamiento lógico es el siguiente. Un hombre posee tres cosas: un lobo, una oveja y una berza, y tiene que transportarlas a la orilla opuesta de un río. Pero el bote que utiliza es muy pequeño y sólo puede llevar una cosa cada vez. Y si deja en tierra el lobo y la oveja, el lobo le matará la oveja, mientras que si deja a la oveja y la berza, la oveja se comerá la berza. ¿Cómo puede hacer para pasar a la otra orilla del río las tres cosas conservándolas intactas?

Este enigma puede resolverse con un ordenador (si te gusta, puedes intentar escribir un programa que lo haga), pero es mucho más rápido hacerlo con papel y lápiz o hacerlo directamente de memoria.

Los tipos de enigmas que mejor se resuelven por ordenador son los que contienen un cierto número de hechos y cifras y en los que se pide encontrar el valor de alguna incógnita. Otros que también funcionan bien sobre ordenador son los que exigen realizar tareas complicadas de tipo aritmético o geométrico. En tales casos es más rápido escribir un programa que resuelva el problema que intentar hacerlo con papel y lápiz.

Este artículo te enseñará cómo resolver tres de los tipos más comunes de *puzzles*. Antes de mirar las soluciones, merece la pena que te entretengas en gastar algo de tiempo intentando resolverlos tú solo. Después, y si no has dado con la solución, pasa a consultar los programas y la explicación de su funcionamiento.

SIMPLIFICACION DEL PROBLEMA

Si no estás muy acostumbrado a resolver adivinanzas, al principio puede que la cosa te resulte bastante difícil, ya que se requiere un poco de práctica para separar la información esencial de la masa de palabras confusas que sólo sirve para despistar. De hecho los enigmas del primer tipo se basan en las palabras que enmascaran lo que con frecuencia es un problema bastante sencillo. Cuando se le despoja de sus palabras, el enigma conduce habitualmente a la solución de un sistema de ecuaciones. Aquí tienes un sencillo problema que puedes resolver sin ordenador y que te ayudará a entender los principios que intervienen.

Un grupo de amigos entra en un café y pide tres cafés y dos tés. La cuenta asciende a 176 pesetas. Al día siguiente se les agrega alguien más, y piden el doble de tés y un café menos. Esta vez la cuenta sube 16 pesetas más. ¿Cuánto cuesta la taza de té?

Si eliminas toda la información no esencial y utilizas letras en vez de palabras, el problema se reduce a $3c + 2t = 176$ y $2c + 4t = 192$.

A esto se le llama un sistema de dos ecuaciones, porque han de satisfacerse las dos al mismo tiempo. Para poder resolver un sistema de ecuaciones se necesitan al menos tantas ecuaciones como incógnitas. En este caso hay dos incógnitas a las que hemos llamado c y t, así como dos ecuaciones, por lo que el sistema en principio tiene solución. Además las ecuaciones son lineales ya que ninguna de las variables aparece elevada a una potencia mayor que 1. Por ejemplo $-3c^2 + 2t = 176$ es una ecuación no lineal que se resuelve por un método diferente.

Para resolver el problema puedes reordenar la segunda ecuación, lo que te dará $c = (192 - 4t)/2$ y a continua-

ción sustituir en la primera. Se obtiene 3(192 − 4t)/2 + 2t = 176. Reagrupando de nuevo los términos, se obtiene t = 28, por lo que una taza de té cuesta 28 pesetas.

Las ecuaciones con dos incógnitas como ésta son fáciles de resolver y las de tres incógnitas tampoco son demasiado difíciles. Pero para un mayor número de incógnitas el ordenador resulta decisivo a la hora de hacer por tí el trabajo más duro.

EL VENDEDOR DE SELLOS

Aquí tienes el primer enigma real que debes resolver.

Un vendedor de sellos tiene una caja con sellos extranjeros que piensa vender en paquetes de seis precios diferentes. El precio de los paquetes está codificado desde la A a la F. Seis jóvenes miembros de un club filatélico se presentan a la vez dispuestos a gastarse todo el dinero que llevan en el bolsillo en lo siguiente:

	A	B	C	D	E	F	TOTAL
Elena	6	2	3	1	1	2	341
Jaime	14	11	0	1	2	1	469
Anita	0	1	3	6	4	3	598
Mar	5	3	5	2	1	1	376
Luis	12	1	4	4	3	2	587
Clara	8	0	1	1	0	3	293

La pregunta es: ¿Cuál era el precio de cada paquete?

El intentar resolver este sistema de ecuaciones eliminando una variable en cada paso, es posible, pero requeriría mucho tiempo y se presta a que se cometan errores de cálculo. Con el siguiente programa podrás resolver esta adivinanza así como cualquier otro sistema de ecuaciones lineales. Esta clase de sistemas se presentan en muchos tipos de problemas y no sólo en la resolución de acertijos.

Existen varias formas de resolver sistemas de ecuaciones como éste, y puedes encontrar métodos diferentes, pero el programa que sigue es rápido y relativamente corto:

```
10 CLS:INPUT"NUMERO DE FILAS"
   ;R
15 INPUT"NUMERO DE COLUMNAS";
   C:PRINT
20 DIM A(R,C),B(R,C),
   A$(C-1)
30 FOR K=1 TO C-1
40 PRINT"INCOGNITA DE LA
   COLUMNA";K;:INPUT A$(K)
50 NEXT:PRINT
60 FOR J=1 TO R
70 PRINT"COEFICIENTES DE LA
   FILA";J
80 FOR K=1 TO C
```



```
90 INPUT A(J,K)
100 B(J,K)=A(J,K)
110 NEXT K:NEXT J
120 FOR L=1 TO R
130 GOSUB 230
140 GOSUB 280
150 NEXT
160 CLS
170 FOR K=1 TO C-1:PRINT
    TAB(8*K-7);A$(K);:NEXT
    :PRINT
180 FOR J=1 TO R:FOR K=1
    TO C:PRINT TAB(K*8-8)
    ;B(J,K);
190 NEXT:PRINT:NEXT
200 PRINT:PRINT
    "RESULTADOS:-"
210 FOR K=1 TO C-1:PRINT
    TAB(8*K-8);A(K,C);:NEXT
220 END
230 REM
240 D=A(L,L)
250 FOR K=1 TO C
260 A(L,K)=A(L,K)/D
270 NEXT:RETURN
280 REM
290 FOR J=1 TO R
300 IF J=L THEN NEXT
    :RETURN
310 F=A(J,L)
320 FOR K=1 TO C
330 A(J,K)=A(J,K)-F*A(L,K)
340 NEXT:NEXT:RETURN
```

La primera parte del programa, entre las líneas 10 y 110, te permite introducir los datos, mientras que la segunda parte, situada entre las líneas 120 y 150 llama a dos subrutinas (o procedimientos) para calcular la respuesta, la cual se imprime entonces en las líneas 160 a 220.

Los valores se cargan en la matriz A(J,K) a razón de una fila cada vez. En este programa, J se refiere a las filas y K a las columnas, lo cual has de tener siempre en la mente al leer el resto del programa. La matriz A(J,K) se copia sobre una matriz idéntica B(J,K) de forma que se pueden imprimir los valores originales junto con la respuesta en la línea 180.

El cálculo avanza de fila en fila, controlado por el bucle que se extiende entre las líneas 120 y 150. Antes de hacer otras cosas, la rutina de las líneas 230 a 270 extrae el elemento diagonal de la fila (es decir A(1,1), A(2,2), etc.) y divide cada elemento de dicha fila por ese valor. El valor del elemento diagonal resultante es ahora igual a uno, es decir, ha quedado normalizado.

La siguiente rutina es la encargada de hacer la mayor parte del trabajo de este programa. Aunque sólo contiene seis líneas, resulta muy difícil examinar lo que hace. La mejor manera de entenderlo es examinar a fondo un ejemplo real (aunque corto) siguiendo uno por uno los pasos que va dando el ordenador. En esencia funciona de la siguiente manera. Supongamos que ya has normalizado la fila 1, por lo que L es 1. La línea 290 toma entonces cada fila diferente de la 1 (línea 300), por lo que en este caso empieza por la 2. La línea 310 toma el primer elemento de la fila 2 y le llama F. Todavía en la fila 2, la línea 320 va recorriendo todas sus columnas, y la línea 330 multiplica F por el elemento correspondiente de cada columna, pero de la fila 1, restándolo del elemento de la misma columna en la fila 2. Esto mismo lo va haciendo con las filas 3, 4, 5 y 6. Seguidamente empieza otra vez con L igual a 2.

Al final de este proceso, los elementos diagonales de cada fila siguen siendo 1 y todos los demás elementos son ceros. Por eso se pueden leer con facilidad los valores en la columna de la derecha.

LOS REGALOS DE NAVIDAD

El programa que acabamos de ver sirve para resolver cualquier sistema de ecuaciones lineales en el que haya tantas ecuaciones como incógnitas. Pero existen rompecabezas en los que las cosas se organizan de tal manera que no hay ecuaciones suficientes, por lo que no se pueden resolver de esta forma. De hecho no existe una solución única. El truco consiste en seleccionar la respuesta más plausible entre todas las soluciones disponibles. Normalmente el enigma contendrá una clave que te ayudará. También podría ocurrir que las ecuaciones resultaran ser no lineales, como en el problema que te vamos a plantear.

Ensaya el siguiente enigma. En Navidad, el **tío Alberto**, que es un matemático un poco excéntrico, explicó a sus dos jóvenes sobrinos que le daría a cada uno tantos paquetes como la edad que tenían (contando sólo los años). También les dijo que cada paquete contenía el mismo número de sobres que la edad de cada uno, y que cada sobre contenía un número de pesetas igual a la edad de cada niño. Después de preparar los regalos, se le oyó murmurar entre dientes que el siguiente año la cosa le iba a costar 500 pesetas más. ¿Qué edades tenían los dos niños?

Suprimiendo las palabras y llamando A y B a las edades de los niños, así como M a la cantidad de dinero gastada en este año, la solución del enigma queda reducida a la de las ecuaciones siguientes:

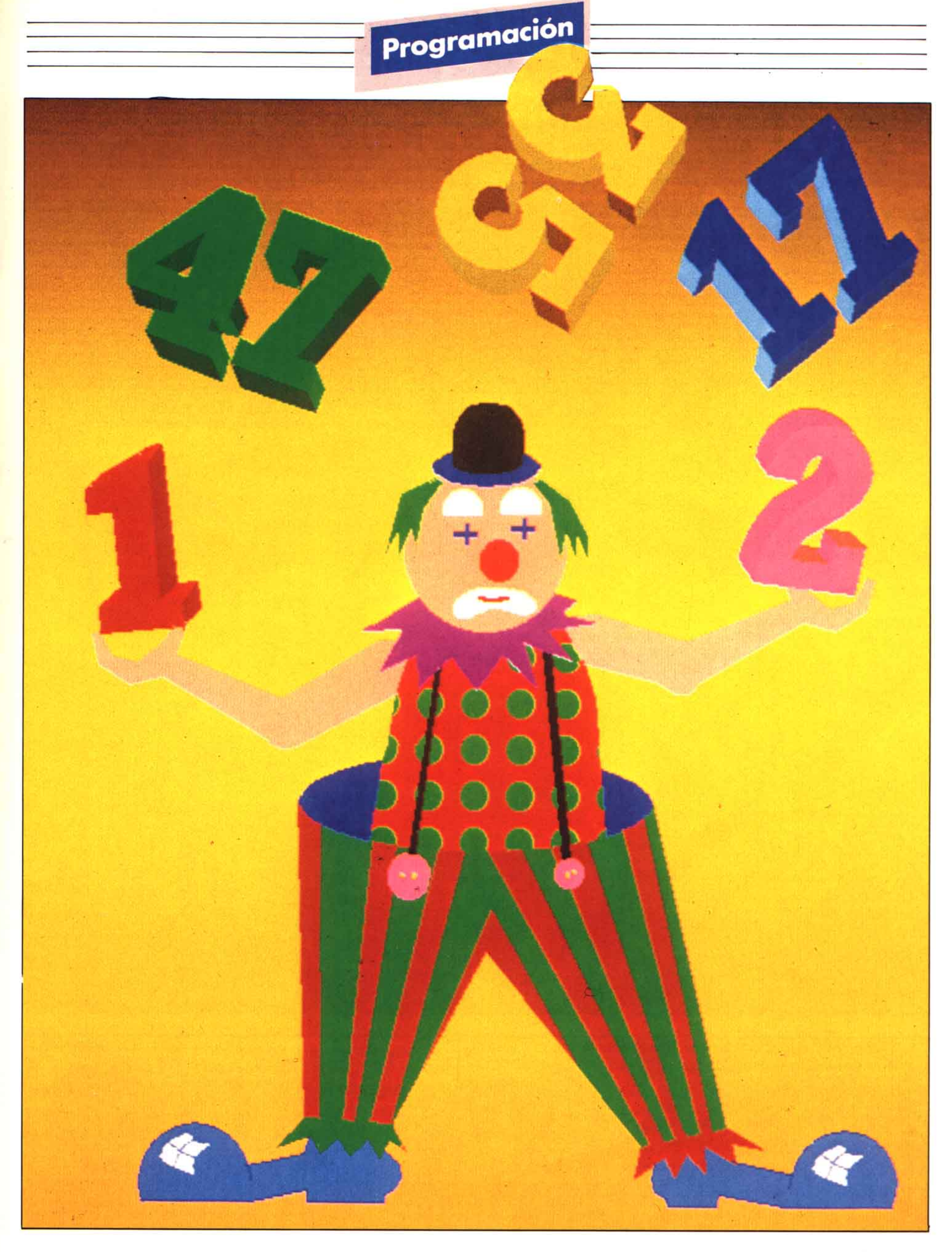
$A \uparrow 3 + B \uparrow 3 = M$;

$(A + 1) \uparrow 3 + (B + 1) \uparrow 3 = M + 500$

Tenemos tres incógnitas: A, B y M, pero sólo dos ecuaciones. Sin tener un ordenador, podrías utilizar un método de prueba y error para ir ensayando diferentes valores de A y B, y observando si las ecuaciones se satisfacen. El ordenador trabaja de una forma parecida, salvo que naturalmente puede hacer varios cientos o miles de ensayos de valores diferentes en un tiempo muy corto y sin cometer errores.

El primer paso en la resolución del problema es buscar en el enunciado del enigma alguna pista suplementaria que te pueda ayudar a limitar el margen de valores que debes ensayar. Como se habla de «jóvenes» sobrinos, no es probable que tengan más de 14 años, y es muy probable que al menos tengan tres años. El programa para resolver el problema es muy corto:

```
10 FOR A=3 TO 14
20 FOR B=A TO 14
30 M=A^3+B^3
40 N=(A+1)^3+(B+1)^3
50 IF ABS (M+500-N)<.01
   THEN CLS:PRINT"LA SOLUCION
   ES:":PRINT "A=";A,"B=";B
   :END
```


47
99
17
1
2

```
60 NEXT B
70 NEXT A
```

Al ejecutar este programa obtendrás solamente una respuesta, debido a las restricciones introducidas en las líneas 10 y 20. Si en el enigma no se hubiera especificado que los sobrinos son jóvenes, el bucle FOR ... NEXT tendría que extenderse más. Digamos de paso que la condición impuesta en la línea 50 evita problemas con los errores de redondeo; lo que realmente se está comprobando es si se cumple que M + 50 es igual a N.

Este método debe funcionar para todos los problemas con más incógnitas que ecuaciones. En otros tipos de problemas, puede que te sea necesario hacer hipótesis acerca de los valores de más variables, o tal vez incluir más condiciones IF ... THEN del tipo de la que aparece en la línea 50. Con un número muy elevado de incógnitas o de condiciones el programa puede tardar muchos minutos o incluso horas en ejecutarse, aunque seguirá funcionando y resolverá, tarde o temprano, el enigma.

ECUACIONES SINGULARES

Hay algunas situaciones especiales, con frecuencia bastante tontas, en que los sistemas de ecuaciones simultáneas tienen o una solución ambigua o carecen por completo de solución. Las ecuaciones $x = 2$ y $x = 3$ obviamente no pueden satisfacerse al mismo tiempo, e incluso cuando tiene tantas ecuaciones como incógnitas, como en el caso $x + y = 3$; $x + y = 1$ no pueden cumplirse simultáneamente.

Otro caso son las ecuaciones $x + y = 2$; $2x + 2y = 4$. En este caso se trata de dos ecuaciones que no son independientes; esencialmente se trata de la misma ecuación, existiendo muchas soluciones posibles.

Cuando cuentes con el número de ecuaciones adecuado y siga habiendo incompatibilidades o las ecuaciones no sean independientes, se dice que el sistema es singular.

El caso de las ecuaciones dependientes se puede resolver utilizando un método de prueba y error, como hacíamos en el caso de los regalos de Navidad. Pero al ser posibles varias soluciones necesitas alguna información adicional para seleccionar la correcta.

MAGIA MATEMATICA

El tercer tipo de rompecabezas que suelen aparecer en las revistas de informática son los problemas de números. Suelen plantear cuestiones de aritmética aparentemente sencillas que resultan extraordinariamente laboriosas de resolver a menos que se utilice un ordenador.

Aquí tienes un par de ejemplos.

Existe un número de cuatro cifras que cuando se invierte y se multiplica por un entero se convierte en el número original: 8712 = 4*2178. El problema es encontrar si existen más números con esta misma propiedad, y en caso afirmativo calcularlos.

Otro número interesante es el 987654321. Este número es un múltiplo exacto de 17 y utiliza todos los dígitos del 1 al 9. Trata de encontrar el número inferior más proximo que tenga estas mismas propiedades.

Las soluciones a este tipo de problemas se basan en considerar el número como una colección de cifras más bien que como un valor numérico. Puedes, o dividir el número en unidades, decenas, centenas, etc, o tratarlo como una cadena de cifras y separar éstas con ayuda de RIGHT$, MID$ y LEFT$.

El primer problema se resuelve mediante el primer método. Si representamos el número por ABCD, da origen a la siguiente ecuación:

$$1000*A + 100*B + 10*C + D = X*(1000*D + 100*C + 10*B + A)$$

cuyas incógnitas tienes que calcular. Como de costumbre la principal dificultad consiste en decidir qué valores se ensayan. A puede variar entre 1 y 9, no pudiendo ser cero ya que en ese caso tendrías un número de tres cifras. B y C pueden variar entre 0 y 9. El factor de multiplicación X debe ser al menos 2 y no debe superar a 10/D, para que el segundo miembro de la ecuación sea un número de cuatro cifras. Por la misma razón D no puede ser mayor que 4. Ahora puedes ya escribir el programa para resolver el problema:

```
10 FOR A=1 TO 9
20 FOR B=0 TO 9
30 FOR C=0 TO 9
40 FOR D=1 TO 4
50 FOR X=2 TO INT(9.9/D)
60 J=1000*A+100*B+10*C+D
70 K=1000*D+100*C+10*B+A
80 IF X*K=J THEN PRINT J;
   "=";X;"*";K
90 NEXT X,D,C,B,A
```



Teclea el programa, a continuación teclea RUN y siéntate a esperar porque llevará bastante tiempo la comprobación de todas las combinaciones posibles.

La solución del segundo problema trata a los números como cadenas.

```
10 M=987654321#
20 M=M-17
30 M$=STR$(M)
40 F=0
50 FOR P=1 TO 9
60 P$=CHR$(48+P)
65 FOR Z=1 TO LEN(M$)
70 IF MID$(M$,Z,1)=P$
   THEN 90
80 NEXT Z:F=F+1
90 NEXT P:IF F=0 THEN
   PRINT M$:END
100 GOTO 20
```

El programa parte de 987654321 y va contando hacia abajo de 17 en 17, convirtiendo cada múltiplo de 17 en una cadena de números a la que llama M$. Las líneas 50 y 60 convierten cada dígito del 1 al 9 en una cadena y a continuación la línea 70 comprueba si dicha cadena figura en M$. Si hay algún dígito que no esté en M$, hay un indicador F que se incrementa una unidad. Sólo se imprime M$ cuando contiene todos los dígitos del 1 al 9.

ECUACIONES SINGULARES

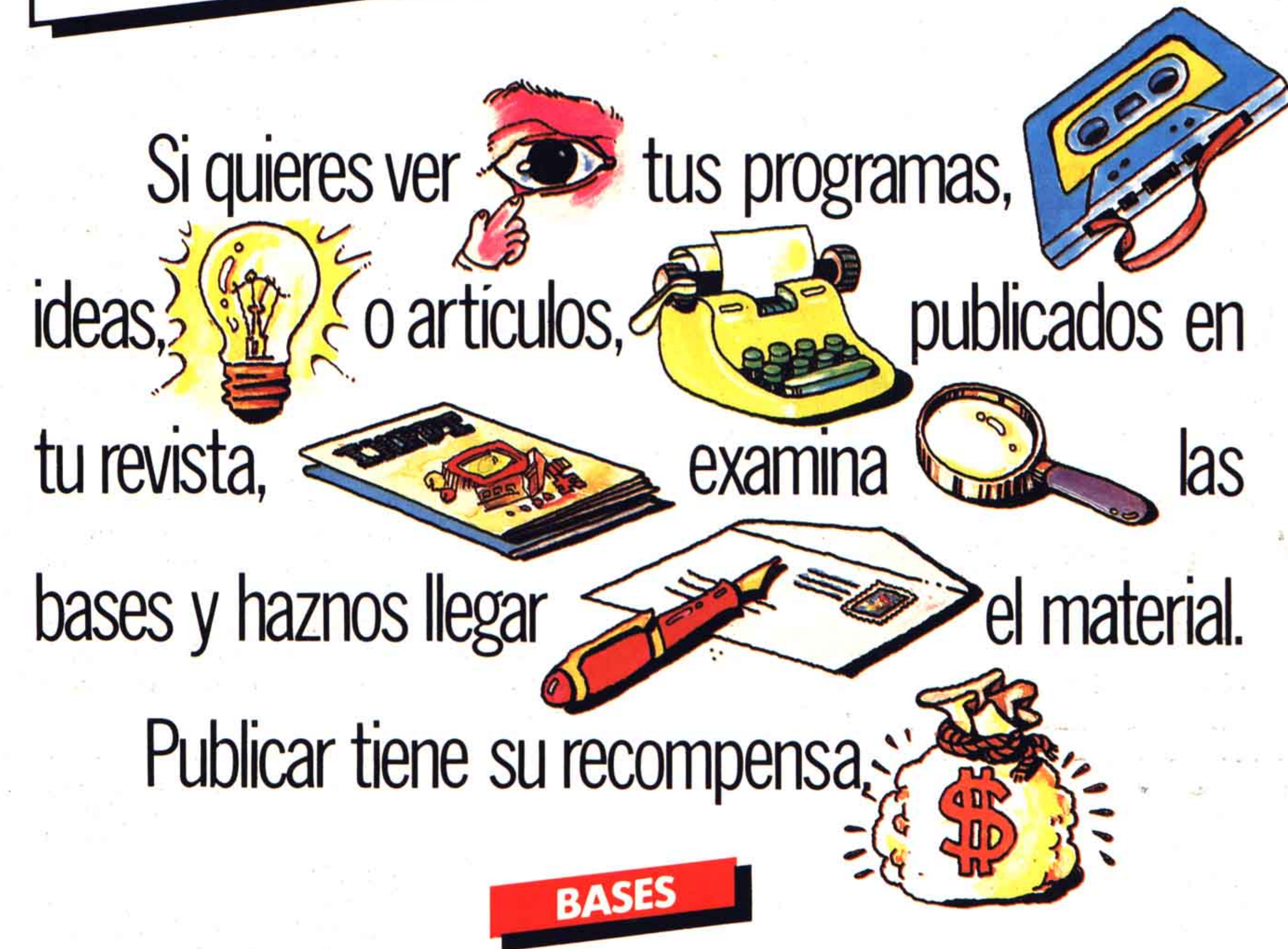
Hay algunas situaciones especiales, con frecuencia bastante tontas, en que los sistemas de ecuaciones lineales tienen o una solución ambigua o carecen por completo de solución. Las ecuaciones $x = 2$ y $x = 3$ obviamente no pueden satisfacerse al mismo tiempo, e incluso cuando tienes tantas ecuaciones como incógnitas, como en el caso $x + y = 3$; $x + y = 1$ no pueden cumplirse simultáneamente.

Otro caso son las ecuaciones $x + y = 2$; $2x + 2y = 4$. En este caso se trata de dos ecuaciones que no son independientes; esencialmente se trata de la misma ecuación, existiendo muchas soluciones posibles.

Cuando cuentes con el número de ecuaciones adecuado y siga habiendo incompatibilidades o las ecuaciones no sean independientes, se dice que el sistema es singular.

El caso de las ecuaciones dependientes se puede resolver utilizando un método de prueba y error, como hacíamos en el caso de los regalos de Navidad. Pero al ser posibles varias soluciones necesitas alguna información adicional para seleccionar la correcta.

¡PARTICIPA EN INPUT!



BASES

PROGRAMAS: Una vez desarrollado tu programa, que debe ser original y no haber sido enviado a ninguna otra publicación, puedes enviárnoslo aquí grabado en cassette, diskette o microdrive. Es preferible que vaya acompañado por un listado de impresora, pero no es imprescindible.

El programa habrá de venir acompañado por un texto que aclare cuál es su objetivo, el modo de funcionamiento y una explicación del cometido que cumplen las distintas rutinas que lo componen. El texto se presentará en papel de tamaño folio y mecanografiado a dos espacios. No importa que la redacción no sea muy clara y cuidada; nuestro equipo de expertos se encargará de proporcionarle la forma más atractiva posible.

ARTICULOS E IDEAS: Se aplica lo anteriormente dicho para los textos que acompañan a los programas; es decir, conviene detallar al máximo lo que desees que aparezca publicado en la revista, de la manera que te gustaría que otra persona hubiera explicado eso mismo.

UN JURADO propio decidirá en cada momento qué colaboraciones reúnen los requisitos adecuados para su publicación, y evaluará la cuantía del premio en metálico al que se hagan acreedoras.

No olvidéis indicar claramente para qué ordenador está preparado el material, así como vuestro nombre y dirección y, cuando sea posible, un teléfono de contacto. Entre todos los trabajos recibidos durante cada mes SORTEAREMOS:

- **Un premio de 50.000 ptas.**
- **Un premio de 25.000 ptas.**
- **Un premio de 10.000 ptas.**

en material microinformático a elegir por los afortunados.

¡No os desaniméis!, por muy simples o complejas que puedan parecer vuestras ideas, todas serán revisadas con el máximo interés.

INPUT MSX
Alberto Alcocer, 46, 4.° B
28016 Madrid

NOTA: INPUT no se responsabiliza de la devolución del material que no vaya acompañado por un sobre adecuado con el franqueo correspondiente.



ESTRUCTURA TUS PROGRAMAS PARTE 2

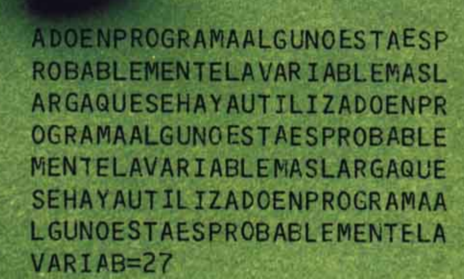
Cuando hayas terminado con el diseño general de un programa, y hayas escrito todos los módulos individuales que lo constituyen, puedes empezar a pensar en la forma de probar los módulos. Después tendrás que ocuparte de la forma en que vas a combinar y ensamblar entre si los distintos bloques.

La mayoría de las subrutinas y módulos necesitarán unirse con el resto del programa de una u otra forma. Esto se hace normalmente utilizando variables. Algunas variables, que se especifican al principio, se pasarán a la rutina constituyendo lo que se llama parámetros de entrada. Hay otras variables que serán introducidas en el programa por la propia rutina, son los parámetros de salida. Es muy importante que las variables se especifiquen de forma precisa.

Cuando empieces a escribir un programa, debes hacer una lista de las variables que necesites usar, junto con una descripción de su uso y sus posibles valores, si los conoces. De no hacerlo así, aunque puede ser que al principio del programa sepas muy bien lo que significa cada letra, es más que probable que se te haya olvidado cuando vuelvas sobre el programa más adelante. Resulta de gran ayuda la utilización de nombres largos de variables, en los casos en que tu ordenador lo permita y siempre que no andes muy escaso de espacio en la memoria.

En el caso de tu **MSX** puedes trabajar con nombres de variables de cualquier longitud, con la única limitación de que no es posible superar, en una línea de programa (a menos que se utilice algún truco) la longitud de 255 caracteres. Teniendo esto en cuenta puedes trabajar perfectamente con una variable como ésta; funciona a la perfección...

```
10 ESTAESPROBABLEMENTELAVARIA
   BLEMASLARGAQUESEHAYAUTILIZ
```



```
ADOENPROGRAMAALGUNOESTAESP
ROBABLEMENTELAVARIABLEMASL
ARGAQUESEHAYAUTILIZADOENPR
OGRAMAALGUNOESTAESPROBABLE
MENTELAVARIABLEMASLARGAQUE
SEHAYAUTILIZADOENPROGRAMAA
LGUNOESTAESPROBABLEMENTELA
VARIAB=27
```

Tiene nada menos que 240 caracteres!

Lo que sí tienes que tener en cuenta, para no llevarte sorpresas, es que tu ordenador sólo considera los dos primeros caracteres del nombre de cualquier variable (por ejemplo no distingue entre las variables VE y VELOCIDAD), así que por muy largo que pongas el nombre, asegúrate de que los dos primeros caracteres sean distintos a los del resto de las variables que utilices.

Aquí tienes un ejemplo de la forma en que podrías especificar las variables en una rutina de ordenación por el método de la burbuja:

rutina de ordenación por el método de la burbuja:

Avanza un paso más en la estructuración de tus programas, tomando como ejemplo la construcción de un programa de ordenación con el que podrás ordenar desde bolas de billar hasta las voces de un diccionario.

- SEGUIMIENTO DE LAS VARIABLES DEL PROGRAMA
- SUBRUTINAS
- COMO PROBAR LOS MODULOS
- PONIENDOLO TODO JUNTO



ordenación de la parte especificada de una matriz según el orden alfabético.

Variables de entrada:

A$(N) matriz unidimensional que se quiere ordenar (valor de N≥1)

N1 primer elemento de la matriz que se quiere ordenar (1≤N1≤N2)

N2 último elemento de la matriz que se quiere ordenar (N1≤N2≤ longitud de A)

Variables de salida:

A$(N) matriz ordenada

Variables temporales:

Z, Z$, I

Es útil anotar las variables temporales utilizadas dentro de una subrutina para evitar conflictos entre los módulos o con el resto del programa. También te será de utilidad reservar algunas letras especialmente para dichas variables temporales. Por ejemplo, podrías usar de Z0 a Z9. Con esto además consigues evitar el derroche de espacio para las variables.

Si organizas bien todos estos aspectos al principio de la construcción de un programa, te ahorrarás muchos quebraderos de cabeza más adelante. Muchos errores de programación se deben a que las variables han sido corrompidas, es decir, han cambiado de valor cuando tú no esperabas que lo hicieran. Pero si sigues el método anterior, no tendrás ningún problema.

La redacción de una lista de variables también te será útil si alguna vez cambias el programa más adelante, ya que entonces será el momento de consultar la lista donde se especifica el modo en que se utilizan dichas variables. Con esa lista te resultará mucho más rápida la modificación de las variables y te evitarás la introducción de nuevos errores en la programación, debidos a la corrupción de otras variables que ya habían sido usadas para otros fines. Acuérdate de anotar las modificaciones que introduzcas, así como la fecha de las mismas.

Aquí tienes ya el listado de la rutina de ordenación por el método de la burbuja. En las páginas siguientes tienes una explicación de su forma de funcionamiento y el correspondiente diagrama de flujo.

```
1000 REM RUTINA DE ORDENACION
     DE UNA MATRIZ POR EL
     METODO DE LA BURBUJA
1005 REM VARIABLES
     (A$(N),N1,N2)
1010 Z=0
1020 FOR I=N1 TO N2-1
1030 IF A$(I)<=A$(I+1)
     THEN GOTO 1060
1040 SWAP A$(I),A$(I+1)
1050 Z=1
1060 NEXT I
1070 IF Z=1 THEN
     GOTO 1010
1080 RETURN
```

El programa anterior es una subrutina a la que hay que llamar desde el programa principal. Por ejemplo, para ordenar los elementos números 5 a 20, se podría llamar así a la subrutina:

```
100 N1=5:N2=20
    :GOSUB 1000
```

Siempre tiene que haber además una corta sección de programa para introducir los elementos que quieras ordenar, y otra sección que presente la lista ya ordenada. Llegados a este punto, tienes que decidir la forma en que quieres que aparezca el resultado en la pantalla, haciendo incluso algún dibujo sobre papel si ello te resulta de ayuda.

PRUEBA DE LOS MODULOS

Cada uno de los módulos de tu diseño original terminará por convertirse en una subrutina de tu programa. Este método de fraccionamiento del programa ayuda mucho durante la



fase de pruebas, ya que cada uno de los módulos puede ser probado y depurado separadamente. La idea básica consiste en definir las variables de entrada, llamar a la subrutina y examinar los resultados. Volviendo a la rutina de ordenación por el método de la burbuja, puedes hacer una prueba como la siguiente:

```
8 CLS:INPUT"NUMERO DE
  ELEMENTOS";N
10 DIM A$(N)
12 CLS:PRINT"INTRODUCE LOS";N
   ;" VALORES"
14 FOR I=1 TO N:INPUT
   A$(I):NEXT I
16 CLS:INPUT"RANGO PARA LA
   ORDENACION (N1,N2)"
   ;N1,N2
18 GOSUB 1000
20 CLS:PRINT
   "LISTA ORDENADA"
22 FOR I=1 TO N:PRINT A$(I)
   :NEXT I
24 FOR J=1 TO 1000:NEXT
   :GOTO 16
```

Un aspecto importante a la hora de construir un programa a base de módulos es incluir en cada uno de ellos lo que puede denominarse un «filtro». No se trata más que de un conjunto de líneas que se encargan de comprobar que los parámetros de entrada al módulo son correctos.

En un programa complejo resultaría imposible comprobar la corrección de todas y cada una de las entradas o salidas de los módulos. El programa se haría insoportablemente lento en su ejecución.

Lo que sí puede y debe hacerse es comprobar al menos los valores límite, de forma que el programa pueda detectar qué variables caen fuera de su rango permitido.

Los filtros se suelen utilizar siempre que, en alguno de los módulos, haya entrada desde el teclado. Un ejemplo típico es el siguiente:

```
1010 INPUT"ESCRIBE UN NUMERO
     (1-99)";N
1020 IF N<1 OR N>99 THEN
     GOTO 1010
1030 RETURN
```

Otro consejo útil. Es una buena idea que, durante la etapa de prueba, todas y cada una de las líneas del programa se ejecuten al menos una vez. Hay muchos errores que sólo se detectan cuando se ejecuta una determinada línea. En el caso de saltos condicionales del tipo IF...THEN, debería hacerse la prueba para los dos posibles valores de la condición; verdadero y falso.

PONIENDOLO TODO JUNTO

Finalmente llega el momento de enlazar todos los módulos y probar el programa como un todo completo. A esto se le llama integración del programa. Si le has dedicado el debido tiempo y cuidado a las anteriores fases del programa, este proceso de integración no debe producirte demasiados quebraderos de cabeza. No obstante si se producen problemas, tienes que volver a comprobar todos los módulos sospechosos y modificarlos si es preciso. Al final tienes que tener un programa perfectamente estructurado que haga exactamente lo que tú quieras.

Para refrescar tu memoria, aquí tienes las reglas que te conducirán a escribir un programa estructurado:

1.- Escribe una descripción general del programa.

2.- Fracciónala en módulos a todos los niveles que sea necesario.

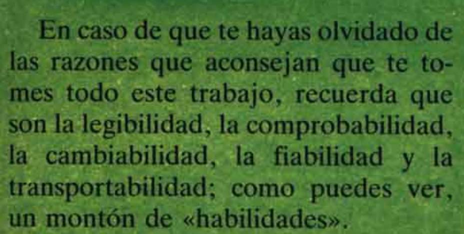
3.- Para cada módulo, dibuja un diagrama de flujo y define las variables de entrada y salida, así como cualquier otro detalle de interés, tal como la forma de presentación en pantalla.

4.- Escribe los programas correspondientes a cada uno de los módulos, utilizando las estructuras descritas en la primera parte de este artículo.

Prueba cada uno de los módulos, dándole unas entradas y comprobando los resultados.

6.- Enlaza todos los módulos y realiza la prueba del programa completo: ¡Tiene que funcionar!





En caso de que te hayas olvidado de las razones que aconsejan que te tomes todo este trabajo, recuerda que son la legibilidad, la comprobabilidad, la cambiabilidad, la fiabilidad y la transportabilidad; como puedes ver, un montón de «habilidades».

Los programas estructurados son más fáciles de seguir por tí mismo y por los demás. Son más fáciles de depurar y de modificar. Es más probable que funcionen bien y resultan más fáciles de adaptar para que corran en otros ordenadores. Si tienes interés en cualquiera de estos resultados, tienes que pasarte a las técnicas estructuradas.

FUNCIONAMIENTO DEL METODO DE LA BURBUJA

Hemos utilizado el método de ordenación de la burbuja para ilustrar la forma en que se puede construir y probar un módulo antes de enlazarlo con el programa principal. Las rutinas de ordenación son muy útiles en cualquier tipo de programas. Esta rutina sirve para clasificar palabras por orden alfabético, pero podría utilizarse igualmente para ordenar números por orden creciente o decreciente. No tendrías más que cambiar la variables Z$ por Z y la matriz A$( ) por A( ).

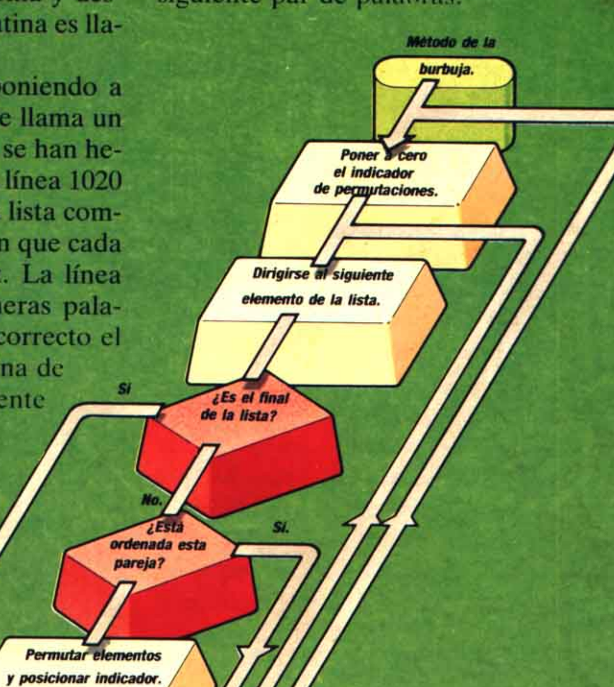
El ordenador lee la lista completa de elementos y los va comparando dos a dos. Si están en el orden correcto los deja como estaban, mientras que si están desordenados los permuta. Continúa haciendo esto por toda la lista haciendo una permutación tras otra hasta que todos los elementos están en el orden correcto.

Para ver mejor los detalles de cómo funciona el programa, conviene compararlo con el diagrama de flujo. La primera parte del programa (que no se refleja en el diagrama de flujo) sirve para encontrar el número de elementos que contiene la lista —N— y definir una matriz llamada A$( ) con espacio suficiente para N elementos. Las líneas 12 y 14 te permiten introducir las palabras, que posteriormente serán ordenadas en la matriz; la línea 16 te pregunta cuáles quieres ordenar. Si quieres ordenar toda la lista, teclea 1, después una coma y después el valor de N. La subrutina es llamada en la línea 18.

La subrutina comienza poniendo a cero el valor de Z. A Z se le llama un indicador (*flag*) y registra si se han hecho o no permutaciones. La línea 1020 crea un bucle que recorre la lista completa. Los números aseguran que cada pareja se compara una vez. La línea 1030 compara las dos primeras palabras y si están en el orden correcto el programa salta sobre la rutina de permutación yendo al siguiente par.

A la línea 1040 sólo se puede llegar si las palabras están en orden incorrecto. En esta línea se intercambian los valores que se están comparando.

Entonces se pone Z a 1 para indicar que al menos se ha hecho una permutación. La línea 1060 envía al programa a comparar el siguiente par de palabras.



Una vez que han sido comparadas todas las parejas, el programa vuelve a la línea 1070. Si Z=1, significa que al menos se ha hecho una permutación, por lo que vuelve a recorrer la lista otra vez. Si no se han hecho permutaciones, es porque la lista ya estaba ordenada, con lo que la subrutina termina y las líneas 20 y 22 presentan la lista ordenada.

MITSUBISHI ML-G1 Y ML-G3

Buscando un puesto en el mercado de los ordenadores profesionales, Mitsubishi **comercializa los modelos** ML-G1 **y** ML-G3**, de la segunda generación** MSX**. Son unas máquinas de excelente acabado y con unas prestaciones que están a la altura de lo que prescribe el estándar.**

Uno de los puntos fuertes del estándar **MSX** es la variedad de equipos compatibles que lo forman. Desde sus inicios han sido muchas las marcas involucradas, lo que ha dado lugar, con los equipos de la primera generación, a un *boom* de modelos, todos ellos compatibles pero cada uno con sus características propias. Algo parecido, aunque a mucha menor escala, está ocurriendo con los **MSX2**. En estas mismas páginas presentamos hace poco los modelos **VG-8235** de **Philips** y **HB-F500P** de **Sony**. Ahora le corresponde el turno a **Mitsubishi**, que presenta dos modelos con aspiraciones: el **ML-G1** y el **ML-G3**.

Con ellos llega una oportuna diversificación de modelos dentro de la nueva versión del estándar, lo que puede ayudar a que éste se difunda y se consolide en un mercado tan complejo como el de la informática personal.

ML-G1

Es el pequeño de la gama **MSX2** de **Mitsubishi** y sigue la filosofía de «incluir todo en una sola carcasa». Este todo está formado por la placa con los circuitos, el teclado y el conjunto de conectores para los periféricos. En la placa están los *chips* de la CPU **Z-80A** (que trabaja al ritmo de un reloj de 3,579545 MHz), de la memoria RAM (64K de RAM principal y 128K de VRAM), de la memoria ROM (32K para el BASIC primera generación, 16K para el BASIC expandido y otros 32K para un programa de aplicación que lleva el nombre de *Melbrains Note*) y por último los *chips* de vídeo, de sonido y los que controlan el funcionamiento de los distintos conectores. El teclado es de «competición». Consta de un bloque principal con el conjunto de teclas alfanuméricas, entre ellas la correspondiente a la letra Ñ, a la que hay que dar la bienvenida y que ha sido necesario incluir para que el sistema cumpliera las normas de homologación. A la derecha de este bloque principal nos encontramos con un teclado numérico independiente (utilísimo a la hora de trabajar con programas tipo hoja de cálculo) y con el conjunto de las teclas de movimiento de cursor. Una fila superior con las teclas de función y el conjunto de te-



ML-G1



ML-G1

clas especiales completan la geografía del teclado. Al tacto resulta inmejorable. No podemos poner, por más que lo intentemos, ni una sola objeción. Es suave, silencioso, todas las teclas resultan perfectamente accesibles y responden a la más ligera presión de





los dedos, sin rebotar ni repetir caracteres. Es un teclado que impresiona y que sólo se puede catalogar como auténticamente profesional.

Por último vamos con los conectores. El primer tanto se lo apunta **Mitsubishi** al incluir la alimentación en el interior de la carcasa, evitando alimentaciones externas que no hacen más que estorbar. Así que no hay (ni falta que hace) conector para la alimentación. Lo que si hay es un buen conjunto de conectores que incluyen los estándar **MSX** (*joysticks*, impresora, *cassette* y 2 cartuchos) a los que hay que añadir las salidas RF, de vídeo y RGB (para conectarse a cualquier tipo de pantalla) y una salida extra de la señal de audio destinada a alimentar equipos de audio externos.

En todo este conjunto hay sin embargo un gran ausente; la unidad de *diskettes*, no incluida en este equipo seguramente por razones de coste, para ofrecer un **MSX2** a bajo precio. Pero no deja de resultar un tanto incomprensible, que un equipo con pretensiones profesionales como éste no lleve incluida al menos una unidad. Hoy en día no tiene sentido utilizar un ordenador con fines profesionales a base de *cassette*. Se hace imprescindible una unidad de *diskette*, que en el caso de este **ML-G1** habrá que adquirir por separado, con los inconvenientes que ello supone.

El equipo incluye todas las características estándar de las máquinas de la segunda generación (que hemos resumido en un cuadro). Entre ellas hay que citar, por su especial atractivo, las del reloj en tiempo real, alimentado en este caso mediante una batería recargable que se carga a través de la red (esto garantiza que no habrá que cambiar de pila). Esta opción permite que cualquier programa conozca, en cualquier momento, la fecha y hora exactas. Las posibilidades que esto sugiere son infinitas. Otro elemento de interés lo constituye la función RAM-*disk* (disco en RAM) que permite trabajar en memoria RAM como si se tratara de una unidad de discos, con la ventaja de trabajar a una velocidad muy superior a la de cualquier unidad de discos. Eso si; al apagar el ordenador, toda la información almacenada en este disco RAM desaparece. Las capacidades gráficas, desde las 80 columnas del modo de texto, hasta la resolución máxima de 512×212 puntos y la paleta de 512 colores, siguen siendo el aspecto más llamativo y actual de las máquinas de esta generación.

ML-G3

Si el **ML-G1** conserva, a pesar de sus características, un cierto aire a ordenador doméstico de la gama media (y la falta de unidad de disco lo confirma), el **ML-G3** se acerca mucho más a lo que puede considerarse aspecto profesional. Consta de una carcasa, de color negro, que incorpora la unidad central, circuitería, conectores

y una unidad de *diskettes* de 3.5 pulgadas, y de un teclado, separado de la carcasa a la que se conecta a través de un cable flexible. El teclado es exactamente el mismo que el del modelo **ML-G1** y comparte con él unas excelentes características.

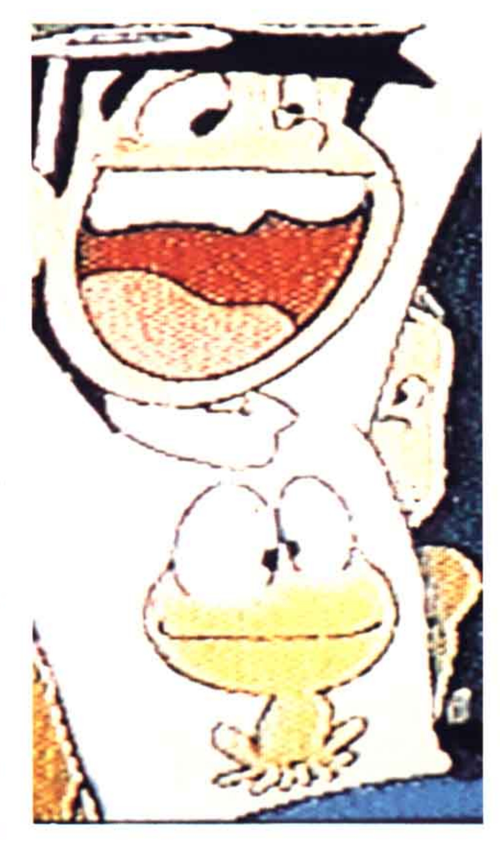
El acabado de la unidad central es casi perfecto; los conectores están todos situados en el panel posterior del aparato, exceptuando los de los *joysticks* y el que conecta con el teclado, que se encuentran situados en uno de los laterales del equipo. En la parte frontal se localiza uno de los conectores de cartuchos, junto a la ranura de la unidad de *diskettes*. En el centro de este panel frontal, oculto por una tapa, hay un alojamiento previsto para incorporación de una segunda unidad de *diskettes* (desde luego una opción muy apetecible). En la parte posterior, además de los conectores mencionados del modelo anterior, se ha incluido un conector para el *interface* serie **RS-232C** incorporado en el equipo (otra opción a tener en cuenta, sobre todo de cara al próximo futuro en el que las comunicaciones jugarán un papel importantísimo en la informática doméstica y profesional).

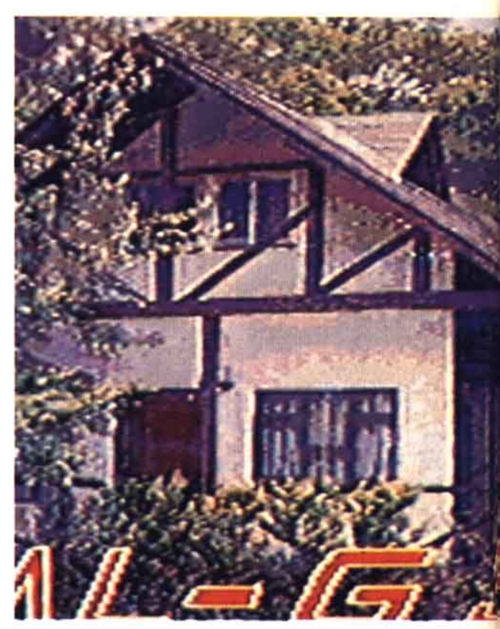
La unidad de *diskettes* de 3.5 pulgadas merece algunos comentarios. Trabaja con dos cabezas de lectura escritura, cada una de las cuales controla la grabación de 80 pistas de 9 sectores cada una y con una capacidad de 512 bytes por sector. Esto proporciona una capacidad total de 720K por *diskette* una vez formateado (el doble que la estándar de las unidades de la primera generación que es de 360K).

La unidad es bastante rápida (es capaz de transferir 250Kbits por segundo) y además es muy silenciosa.

Sólo hay una pega, que en determinados casos puede tener su importancia, y es que el formato de grabación no es compatible con el de la primera generación. De esta forma aunque tengamos programas en *diskette* que puedan funcionar perfectamente en el ordenador, no podremos utilizarlos a menos que hagamos previamente una conversión de formatos de *diskette*.

Otra de las novedades de este equipo es el ya mencionado *interface* serie **RS-232C**. Se trata de un auténtico **RS-232C**, que configura al ordenador como **Equipo Terminal de Datos**. El conector es el estándar **DB-25** de 22 patillas en el que están representadas todas las señales importantes para la conexión.









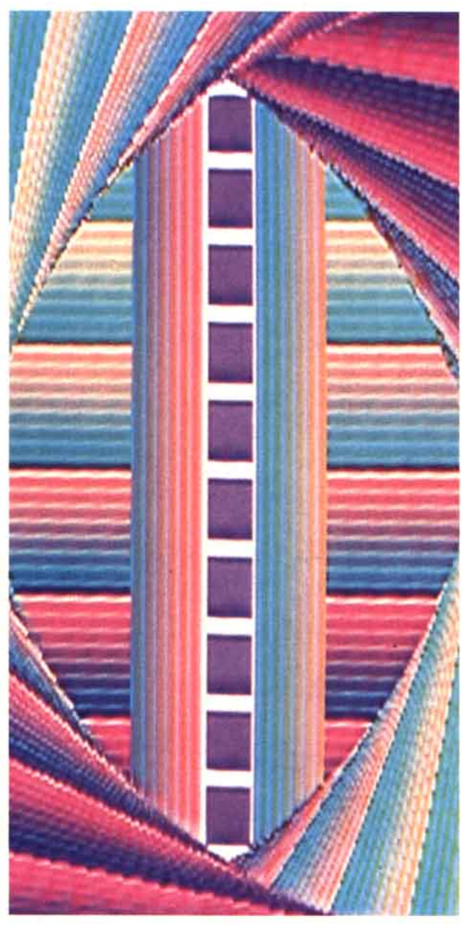












Un par de *chips* (**8251** y **8253**)controlan la conexión y permiten la transmisión de caracteres de 5 a 7 bits, con paridad seleccionable a voluntad y a velocidades que pueden variar entre 50 y 19200 baudios (la velocidad puede ajustarse en valores distintos para transmisión y recepción).

Un tema, en el que seguramente habrá bastantes expectativas, es el del tratamiento de imágenes y señales de vídeo. Existe un sistema actual de **Mitsubishi** para este cometido, pero por desgracia está adecuado al estándar de TV **NTSC** y no al sistema **PAL**. Aunque se habla de una pronta adaptación del sistema a las normas de TV europeas, por el momento no hay otra solución que la de utilizar una cámara de vídeo y un monitor **NTSC**.

Conectando estos dos elementos al digitalizador, que por el momento no viene incluido en los equipos, se podrá trabajar con imágenes de vídeo de la misma calidad que las que reproducimos en este artículo (pertenecientes a uno de los *diskettes* de demostración de **Mitsubishi**).

SOFTWARE

Ambos equipos incluyen de origen (en ROM en el modelo **ML-G1** y en *diskette* en el **ML-G3**) un programa que lleva el nombre de *Melbrains Note*. Aún no hemos tenido ocasión de trabajar con este *software* que al parecer consta de los clásicos procesador de texto, hoja de cálculo, base de datos, gráficos y comunicaciones. Esperamos ofreceros una valoración de este paquete el mes que viene.

El caso es que apenas hay más, por el momento. **Mitsubishi** no ha comentado el lanzamiento de ningún *software* específico que arrope a estas nuevas máquinas, que por el momento tendran que utilizar el de la primera generación, y esto con reservas pues hay unos cuantos programas que se resisten a cargar en las máquinas de la segunda generación aún funcionando perfectamente en las de la primera. Esto nos lleva a una consideración, elemental ciertamente, pero que no siempre se tiene en cuenta y es que por muy bueno que sea el *hardware* (y en este caso es excelente para el nivel de precio considerado) las máquinas sólo pueden llegar a ser verdaderamente interesantes cuando van arropadas por una buena cantidad de programas.

```
                 CARACTERISTICAS DEL ML-G1
                 -------------------------
CPU:                   Z80A  (3.579 MHz)

MEMORIA:               32K  ROM  (BASIC PRIMERA GENERACION)
                       16K  ROM  (BASIC EXPANDIDO)
                       32K  ROM  (MELBRAINS NOTE)
                       64K  RAM  (PRINCIPAL)
                      128K  RAM  (DE VIDEO)

MODOS GRAFICOS:         0   TEXTO   MAXIMO 24 LINEAS X 80 COLUMNAS
                        1   TEXTO   MAXIMO 24 LINEAS X 32 COLUMNAS
                        2   GRAFICO 256 X 192 EN 16  COLORES
                        3   GRAFICO  64 X  48 EN 16  COLORES
                        4   GRAFICO 256 X 192 EN 16  COLORES (DE ENTRE 512)
                        5   GRAFICO 256 X 212 EN 16  COLORES (DE ENTRE 512)
                        6   GRAFICO 512 X 212 EN  4  COLORES (DE ENTRE 512)
                        7   GRAFICO 512 X 212 EN 16  COLORES (DE ENTRE 512)
                        8   GRAFICO 256 X 212 EN 256 COLORES SIMULTANEOS

CHIP DE VIDEO:          V9938       CHIP DE SONIDO:  EQUIVALENTE AL AY38910

CONECTORES:             2 CONECTORES DE CARTUCHO (50 TERMINALES)
                        IMPRESORA, JOYSTICKS
                        RADIO FRECUENCIA (TV), VIDEO COMPUESTO, RGB
                        CASSETTE, SALIDA DE AUDIO EXTERNA


                 CARACTERISTICAS DEL ML-G3
                 -------------------------
MEMORIA:               32K  ROM  (BASIC PRIMERA GENERACION)
                       16K  ROM  (BASIC EXPANDIDO)
                       16K  ROM  (DISK BASIC)
                        8K  ROM  (RS-232C)
                       64K  RAM  (PRINCIPAL)
                      128K  RAM  (DE VIDEO)

UNIDAD DE DISKETTES:       CAPACIDAD:          1MBYTE  SIN FORMATEAR
                                           720KBYTES FORMATEADO
                           VELOCIDAD:      250KBYTES POR SEGUNDO
                           FORMATO:        3.5 PULGADAS
                                           DOBLE CARA (DOS CABEZAS)
                                           80 PISTAS/CABEZA 9 SECTORES/PISTA
                                           512 BYTES/SECTOR
INTERFACE RS-232C:         VELOCIDAD
                           DE TRANSMISION: 50-19200 BAUDIOS
                           BITS/CARACTER:  5,6,7,8
                           BIT PARIDAD:    PAR, IMPAR O NINGUNA
                           LSI EMPLEADOS:  8251 Y 8253
```



LOS MEJORES DE INPUT MSX

PUESTO	TITULO	PORCENTAJE
1.°	Knight Lore	21,3 %
2.°	Soccer	16,4 %
3.°	H.E.R.O.	14,7 %
4.°	Alien 8	11,9 %
5.°	Profanation	9,6 %
6.°	Yie ar kung fu	6,8 %
7.°	Hyper rally	5,6 %
8.°	River raid	5,3 %
9.°	Jet fighter	4,5 %
10.°	Bounder	3,9 %
		100 %

Para la confección de esta relación únicamente se han tenido en cuenta las votaciones enviadas por nuestros lectores de acuerdo con la sección «Los Mejores de Input».

Octubre de 1986.



ABEJORROS SIDERALES

Nuestras naves deben defender nuestra galaxia de los ataques de las malvadas naves enemigas. Ellos tienen todas las de ganar, puesto que son más y más poderosos; sin embargo, no cuentan con nuestra experiencia como pilotos, curtidos en mil combates como éste.
El escuadrón enemigo está formado por tres clases de naves, los cazas azules, muy rápidos y de gran movilidad, a los que podremos abatir de un solo disparo (si somos lo suficientemente rápidos para ello, claro); el platillo volante verde, de escasa movilidad pero de gran capacidad de disparo. Su campo de fuerza protector hace que debamos dispararle tres veces para abatirlo. Por último, tendremos que vérnoslas con la nave insignia. Esta nos atacará transcurrido un tiempo fijo desde que comienza el juego. Es completamente inmune a nuestros disparos, y la única forma de acabar con ella es hacer uso de la bomba. Pero para ello nuestra nave deberá contar con una suficiente carga de fuerza. Cada vez que derribamos un caza azul o un platillo volante verde, dejan libre su carga. Si nosotros la capturamos, aumentaremos la nuestra propia, hasta tener la suficente para poder atacar a la nave insignia antes de que ésta se dirija hacia nosotros. En caso contrario estaremos perdidos, ya que los disparos no la afectan en absoluto. Cuando acabemos con la nave insignia, accederemos a una segunda fase del combate. En ella nos atacarán las naves de la fase anterior, una tras otra, en ráfagas. Después tendremos que aterrizar con nuestra nave en un crater. Luego accedremos a una nueva fase de combate, con nuevas naves y con mayores dificultades. Y así sucesivamente... Hay que hacer mención especial del sonido de este programa. Aparte de los efectos de música y de explosiones, hay varias frases de voz sintetizada en inglés que impresionan por su claridad y realismo. Un verdadero prodigio de voz sintetizada con ordenador.
El juego es divertido, dinámico y difícil. Aun siendo un clásico juego de marcianitos, sin excesivas novedades, la variedad de sus pantallas y su acabado casi perfecto, le convierten en una buena pieza para cualquier cazador de buenos programas. Verdaderamente, pone a prueba nuestra habilidad como pilotos de naves espaciales de combate. Pero seguro que saldremos airosos de la batalla.



DATOS GENERALES

TITULO Valkyr

FABRICANTE Gremlin Graphics

CLASE DE PROGRAMA Juego

FORMATO Cassette

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	8
INTERES	8
GRAFICOS	8
COLOR	9
SONIDO	9
TOTAL	42





****

EL PORTAVIONES

Flight Deck es un programa con características muy interesantes que le situan más allá de lo que es común en juegos de ordenador. En primer lugar porque emplea una serie de dibujos digitalizados para construir los gráficos de determinadas partes del programa, entre ellas la pantalla de presentación. Con ello se consigue una calidad gráfica excelente, poco aprovechada sin embargo, pues la pantallas y áreas gráficas de mayor interés a la hora de jugar (es decir las zonas por las que se mueve el jugador) son

tratadas de forma convencional, con unos gráficos de mediana calidad. Además de la digitalización de imágenes, el programa hace uso, de cuando en cuando, de una serie de digitalizaciones esta vez de voz. Un par de frases suenan por el altavoz del televisor a la hora del despegue de los aviones desde la cubierta del portaviones.
El argumento es bastante clásico: hay que llevar a cabo un ataque contra la base enemiga (situada en una isla) para destruir a las tropas allí situadas. Para ello el jugador



tiene que controlar las evoluciones de los aviones y del portaviones a través de tres fases distintas, que se corresponden con distintas pantallas.
La primera fase es la del despegue de los aviones. La segunda se desarrolla sobre un mapa y en la misma hay que dirigir adecuadamente todas las naves hacia el objetivo. La última fase tiene

DATOS GENERALES

TITULO Flight Deck

FABRICANTE Bytebusters

CLASE DE PROGRAMA Combate aereonaval

FORMATO Cassette

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	8
INTERES	7
GRAFICOS	9
COLOR	9
SONIDO	6
TOTAL	39

lugar sobre la isla; es el ataque final. Este conjunto de fases proporciona la necesaria variedad de escenarios y situaciones para hacer el programa entretenido.
Aunque técnicamente está bastante bien realizado, hay que decir que desilusiona comprobar que las digitalizaciones se utilizan más que nada para conseguir una buena presentación. Otros aspectos, quizá





más imporrtantes, como la velocidad, la respuesta a los controles, los gráficos de las pantallas de acción e incluso el interés general del programa quedan un tanto deslucidos al lado de unos gráficos de presentación tan sumamente excepcionales.

★★★★★★★★★★★★★★★★★★★★★★★★★★★★★

PALEOLITICO SUPERIOR

¿Conseguirá **Thor** culminar con éxito su búsqueda del sentido de la vida? Este se encuentra perdido en las **Montañas del Misterio**. Allí **Thor** tendrá que enfrentarse a múltiples peligros, el mayor de los cuales es el terrible monstruo de las montañas **Grog**.
En su periplo, **Thor** tendrá que recoger cuantas almejas pueda, ya que son la moneda con la que pagar a **Peter**, avispado negociante de la edad de piedra, e inventor del peaje. Para llegar al puente de **Peter**, paso obligado en la búsqueda, hay que pasar por las cuevas oscuras en las cuales, **Thor** deberá orientarse con ayuda de un faro, de invención suya (y precursor de las actuales linternas). Dentro de la cuevas también podrá recoger almejas, pero ¡cuidado con las estalactitas!
Cuantas más almejas recojamos, más ricos seremos para poder negociar con **Peter**, ya que además del peaje del puente, **Peter** es el único suministrador de ruedas en estos tiempos, y nos cobrará 25 almejas por una de repuesto.
Tendremos que guiar a nuestro troglodita a través de los tortuosos caminos de la montaña, esquivando los baches y las rocas. Además, los









DATOS GENERALES

TITULO Grog's Revenge

FABRICANTE U.S. Gold S.A.

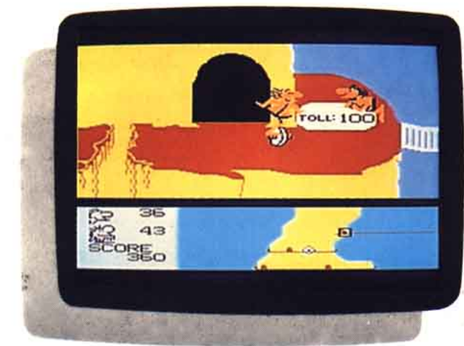
CLASE DE PROGRAMA Juego

FORMATO Cassette

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	10
INTERES	9
GRAFICOS	9
COLOR	10
SONIDO	8
TOTAL	46

almejas, irá directamente a por nosotros.
No cabe duda de que nos encontramos ante uno de los juegos más originales y divertidos que se han diseñado para **MSX**. La calidad de los gráficos y del color es excepcional, pareciendo realmente de dibujos animados.
Hay tres montañas, en cada una de las cuales hay varios niveles de dificultad. El modo para efetuar el



Tiredáctilos están hambrientos, y su principal alimento son las ruedas de piedra asi que ¡mucho cuidado con ellos!
Pero el principal peligro es el terrible **Grog**. Si aparece en la misma pantalla que **Thor**, le destruirá con su terrible grito que hace temblar toda la montaña. Debemos evitar a **Grog** a toda costa, y al mismo tiempo, ser rápidos recogiendo almejas, ya que a él también le gustan para comer (con concha y todo), y cuantas más se coma, menos nos quedarán a nosotros. Además, si **Grog** termina de coger todas las

cambio de nivel muchas veces tendremos que averiguarlo nosotros mismos, y este es otro de los alicientes del juego.
En resumen, un juego dinámico, entretenido y absorbente, y uno de los mejores que han pasado por nuestras manos.

****

DATOS GENERALES

TITULO International Karate

FABRICANTE Endurance Games

CLASE DE PROGRAMA Juego de Karate

FORMATO Cassette

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	7
INTERES	9
GRAFICOS	7
COLOR	7
SONIDO	9
TOTAL	39

KARATE INTERNACIONAL

Te enfrentas con un experto karateka en un torneo por el gran campeonato mundial, que se celebra en diversos paises del mundo.
En cada país, celebrarás un combate a tres asaltos, y ganando dos de ellos podrás pasar al siguiente combate en otro país distinto.
Los asaltos duran un máximo de treinta segundos, a no ser que antes uno de los contrincantes sea derribado por su oponente. El juez, un anciano e imparcial maestro de karate, decidirá si el golpe vale medio punto (**waza-ari**) o un punto (**ippon**). El primero de los rivales que consiga dos puntos, habrá ganado ese asalto.
Combinando los movimientos y el botón del *joystick*, podrás proporcionar una gran variedad de golpes de puño, golpes de pierna, saltos, esquivas, etc., que dan al combate un sorprendente realismo.
Puedes elegir entre luchar con el ordenador, o con otro jugador. En el caso de hacerlo con el ordenador, ten cuidado, pues es un consumado cinturón negro que no te dará tregua ni descanso a lo largo del combate, y

tendrás que hacer uso de todas tus habilidades para derrotarle.
Los efectos sonoros son muy buenos, destacando la voz sintetizada del maestro juez, que da la orden de comienzo y final del combate y marca el valor de los golpes.
El efecto de realismo en los golpes y movimientos es muy bueno, pero el diseño de los karatekas es a nuestro juicio demasiado sencillo.





★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★

EL GRANJERO DE TRANSILVANIA

Tras un complejo y enrevesado argumento (que podemos leer en la parte trasera de la carátula) en el que nos cuentan la historia de un granjero transilvano, de nombre **Fred**, que después de haber perdido la cosecha, se dedica a recoger fruta almacenada en un cercano castillo, armado nada menos que de una pistola láser, se esconde un programa de características poco relevantes.
El personaje (**Fred**) tiene que desplazarse por diversas salas recogiendo frutas. En cada sala hay



además una llave, necesaria para abrir la puerta que da paso a la sala siguiente. Para estorbar el trabajo de recolección de **Fred**, una serie de alimañas (arañas, escorpiones, y otros seres de aspecto extraño) se mueven de forma aparentemente aleatoria por los pasillos de cada sala.
El programa aporta pocas novedades. No hay mapa ni continuidad entre unas salas y otras, por la sencilla razón de que una vez que se accede a la sala siguiente no hay forma de volver atrás. Esto significa que no hay *scrollings*. El paso de una pantalla a la siguiente se resuelve borrando y volviendo a dibujar. La diferencia entre pantallas es pequeña (otro color y algo más de complejidad). Con esta escasez de variedad el juego resulta bastante monótono. Los gráficos de los escenarios y personajes son sólo pasables. El movimiento, la



velocidad y la respuesta a las acciones del jugador tienen un inevitable sabor a programa antiguo, de los que estaban de moda hace algunos años y en ningún modo son representativos de lo que se puede ver actualmente en programas para **MSX**.



DATOS GENERALES

TITULO Forbidden Fruit

FABRICANTE Mind Games

CLASE DE PROGRAMA Arcade

FORMATO Cassette

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	5
INTERES	6
GRAFICOS	7
COLOR	8
SONIDO	6
TOTAL	32



Con todo ello, el programa sólo resulta aceptable si pensamos en el público infantil, en los más pequeños; es sencillo de manejar y tiene un colorido agradable.

★★★★★★★★★

EL REY DE LOS CIRCUITOS

DATOS GENERALES

TITULO Speed King

FABRICANTE Mastertronic

CLASE DE PROGRAMA Juego

FORMATO Cassette

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	7
INTERES	8
GRAFICOS	7
COLOR	8
SONIDO	7
TOTAL	37







¡Sumérgete con este juego en el trepidante mundo de las carreras de motos! Tendrás que mostrar tu habilidad de consumado piloto en una carrera de alta velocidad contra 19 expertos corredores.
El campeonato mundial consta de 10 carreras en diferentes circuitos de todo el mundo, cada uno de ellos con 3 niveles de dificultad.
Encontramos complejas curvas, que pondrán a prueba nuestros nervios y sangre fría y largas rectas, en las que nuestra poderosa máquina desarrollará toda su velocidad.
El cambio de marchas consta de 6 velocidades, y no es fácil escoger la más adecuada en función de las características de la pista.
Al empezar el juego escogeremos circuito y nivel de dificultad, y podremos efectuar unas vueltas de práctica para familiarizarnos con él.
La sensación de movimiento y de velocidad está muy bien conseguida, pero los gráficos podían haberse mejorado mucho en calidad, ya que el dibujo de las motos no es real ni dinámico.
¡Buena suerte, y a por el campeonato del mundo!

****

EL VIAJERO DEL TIEMPO

Magic Knight ha sido catapultado hacia un lejano futuro. Allí, una curiosa criatura de metal, llamada **Klink**, le ha suministrado un **Datacubo**, en cuyo interior se hayan almacenados todos los conocimientos y adelantos técnicos a los que la humanidad ha llegado en el siglo 25.



Magic Knight irá absorbiendo los conocimientos con cuidado de que estos no le causen un grave *shock*, deberá regresar con ellos a nuestro tiempo, para que nuestra civilización pueda beneficiarse de los mismos.
Pero para regresar, nuestro héroe tendrá que encontrar a los guardianes del tiempo, y resolver sus enigmas para obtener de ellos la máquina del tiempo que le permitirá regresar a nuestra era.
No es tarea fácil, y para conseguirla, **Magic Knight** habrá de desenvolverse entre los múltiples peligros de ese tumultuoso siglo e incluso, a veces realizar un viaje espacial.
A lo largo de su aventura, irá encontrando objetos tales como armas, alimento, planos, etc., que podrá examinar, recoger, usar o dejar, según sus necesidades de cada momento.
No dudamos de que con nuestra ayuda, conseguirá llegar sano y salvo a nuestro tiempo y proporcionarnos

los adelantos de un remoto futuro.
El juego es entretenido, pero tiene el defecto de ser muy parecido a otros juegos aparecidos con anterioridad, con lo cual pierde el aliciente de la originalidad, uno de los principales atractivos de cualquier juego.







DATOS GENERALES

TITULO Knight Tyme

FABRICANTE Mastertronic

CLASE DE PROGRAMA
Juego

FORMATO Cassette

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	**6**
INTERES	**8**
GRAFICOS	**7**
COLOR	**8**
SONIDO	**7**
TOTAL	**36**





****

EL TRUENO AZUL

¡Infíltrate con tu super-helicóptero en el territorio enemigo y lleva a cabo la peligrosa misión de rescate! Para lograrlo tendrás que convertirte en un experimentado piloto de un sofisticado helicóptero de combate dotado de las más poderosas armas, bombas y misiles; tendrás que esquivar los peligrosos misiles teledirigidos guiados por calor que te buscarán implacablemente y





evitar los mortíferos dísparos de las naves enemigas.
Para culminar con éxito tu misión habrás de destruir la mayor cantidad posible de armas, carros de combate y fortificaciones en el territorio enemigo. Los depósitos de combustible enemigo que destruyas harán amuentar tu propio suministro. ¡Ten cuidado con acercarte demasiado a tierra o a las fortificaciones enemigas, o serás destruido!
El juego da a su comienzo 4

DATOS GENERALES

TITULO Super Cobra

FABRICANTE Konami

CLASE DE PROGRAMA
Juego

FORMATO Cassette

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	**7**
INTERES	**8**
GRAFICOS	**8**
COLOR	**8**
SONIDO	**7**
TOTAL	**38**

posibilidades, según escojamos 2 ó 4 jugadores y la opción de *joystick* o teclado. El movimiento y los gráficos son estupendos (en la línea de todos los programas **Konami**) y hay que destacar la «mala idea» de los misiles enemigos, con los que tendremos que desarrollar todas nuestras habilidades de piloto experimentado. Un juego entretenido y bien hecho, aunque su diseño diste mucho de ser original; no puede dejar de recordarnos al famoso **Scramble**, por lo menos en cuanto a su idea básica.

****

¿HOMBRE O MOLECULA?

Estamos seguros de que cuando el lector vea las fotos de este juego reconocerá otros de factura sumamente parecida.

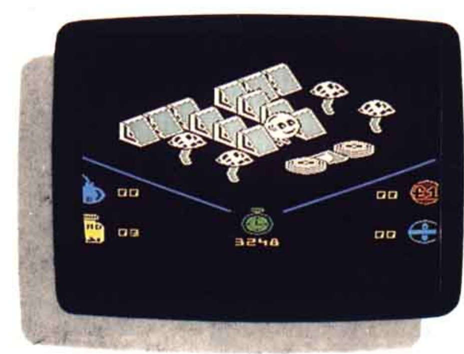


En esta ocasión se trata de salvar a **Molecule Man**, que se encuentra perdido dentro de un complicado laberinto. Nuestros mortales enemigos son el tiempo y la radiación, que irán agotando nuestra vitalidad poco a poco, hasta acabar con nosotros. La única defensa contra la radiación son las pastillas que se encuentran escondidas a lo largo de todo el laberinto. Estas



pastillas prolongarán nuestro tiempo de vida. Pero para conseguir las píldoras tendremos que disponer de las monedas que se encuentran también escondidas.

Con estas monedas también podremos adquirir bombas, que nos



DATOS GENERALES

TITULO Molecule Man

FABRICANTE Mastertronic

CLASE DE PROGRAMA Juego

FORMATO Cassette

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	**5**
INTERES	**7**
GRAFICOS	**9**
COLOR	**6**
SONIDO	**6**
TOTAL	33

servirán para llegar a lugares del laberinto que, de otra manera, serían inaccesibles.

La única forma de que **Molecule Man** pueda escapar del laberinto es que reuna los 16 circuitos necesarios para activar el teleportador que es la única vía de salida. ¡Pero primero tendrá que encontrarlos!

Los gráficos 3D del juego están muy bien conseguidos, y serían muy llamativos si no hubieran sido usados ya en otros muchos juegos del mismo tipo.

Por ello **Molecule Man** no es un juego original en absoluto, aunque si puede hacernos pasar unos ratos entretenidos, si bien no hay duda de



que la repetición de la idea original pueda hacer que nos cansemos antes de él.

****



EL ZOCO

Cambio más de 60 juegos comerciales (Profanation, Sorcery, Knight Lore, Soccer, Tennis...), y cassette especial para ordenador, por unidad de disco de 3.5 pulgadas.

Ramón Alonso
Buenavista, 2B, etl.4ª
Tel. (93) 564 39 06
Montcada i Reixac (Barcelona)

Compro, vendo, intercambio toda clase de juegos/programas educativos en MSX, incluso en disco. Llamar/escribir:

Juan Mantencio Piulachs
Marina, 292, 2º, 1ª
Tel. (93) 255 12 92
08025 Barcelona

Necesito la cinta-programa de la Tableta Gráfica de Spectravídeo: SVI-105, o una copia de la misma, pues no la encuentro en ningún sitio, compensaré y gratificaré.

Carlos Martínez Martínez
Callejón del Moro, 1
Huete (Cuenca)

Cambio urgentemente un ordenador personal Spectrum 48K Plus por un MSX o 28.000 ptas. En caso de cambio por MSX se regalan cables, 15 juegos (Comando, Zorro, Panama Joe, Krazy Kong, etc...). Se admiten de cualquier memoria.

Juan José Escudero
España, 25
San Agustín (Almería)

¡Oferta única! Cambio Yie-ar Kun-fu I y II, Tennis y Futbol de Konami pasados a cinta por cualqueir cartucho de software. Preferiblemente de Pascal o Ensamblador. Interesados ponerse en contacto con:

Juan Carlos Orós
Gasómetro 44, 1º 3ª
Tel. (977) 21 28 40
43001 Tarragona

Cambio 18 programas cartucho, mayoría Konami, por cartucho ampliación de memoria de 64 Kb. Llamar o escribir a:

José Manuel Vera Vilches
Escritor José de los Heros, 3
Tel. (957) 25 47 24 ó 27 48 67
14014 Córdoba

Vendo Hit Bit de Sony HB-55P (32 Kbytes de RAM), ampliación de memoria (64 Kbytes) y 14 interesantes juegos; Hero, Turmoil, Zaxxon, Chess, Gun Fright, Sorsery, Simulador de vuelo, Binary Land, Keysstone Kapers, Ensamblador/Desensamblador, Les Flics, Ninja). Todo por 50.000 ptas, si te interesa llama o escribe a:

David Butxaca Gros
Serra Casa en Pons, 48
Tel. (93) 821 15 40
08600 Berga (Barcelona)

Vendo ordenador Sony HB 55P MSX. Con cables, manual en español y caja de embalaje original. Comprado en navidades y completamente nuevo. El precio es de 15.000 ptas, por razones económicas.

Daniel Díaz Sañudo
Tel. (954) 70 56 33
Sevilla

Cambio cintas de juegos, poseo: Ghostbusters, H.E.R.O., simulador de vuelo, Ninja, etc. Precio a convenir. También desearía comprar el programa ZAXXON.

José Casero Fernández
Soto del Rey, 24
Tel. 893 00 47
Ciempozuelos (Madrid)

Vendo libros «Hacia la Inteligencia Artificial con Amstrad» y «Música y sonidos con Amstrad» de Jeremy Vine, los dos por 1.500 pesetas. ¡Es una ganga!

Vicente Uceda Alvarez
Casiopea, 21
Tef. 682 16 76
Getafe (Madrid)

Vendo ordenador Philips VG-8010. Más 3 manuales y unos 30 juegos comerciales. Con 3 meses de garantía. Precio a convenir. Escribir a:

José Enrique Salvador
Pesquera, 22
Tel. (988) 12 28 86
Aguilar de Campoo (Palencia)

¡Oferta única! Por compra de un MSX 2, vendo ordenador Philips MSX VG-8020 80K memoria (Teclado Profesional), poco uso, con manual BASIC y cables de conexión, sólo por 40.000 ptas.

Carlos Piñeiro Rodriguez
Santo Domingo, 23
Cangas (Pontevedra)

¡Oferta Sensacional! Cambio Knignt Lore, Hero, Time Bandits, Gyro Adventure, Boulder Dash, Zaxxon por cartucho de ampliación de 64 Kb. Llamar o escribir a:

Laureano Lama Pérez o José Carlos
Arboleda, 59
Tel. (954) 77 23 36 de 10 a 1 de la mañana
41130 La Puebla del Río (Sevilla)

Cambiaría programas Base de Datos Databa 32 K-MSX y Procesador de Textos Compor 32 K-MSX, nuevos por otros programas de gestión o didácticos.

J. García
Apartado 20
28080 Madrid

Cambio juegos de cinta: Samuray Ninja, Hero, Zaxxon, Disc Warrior, Billar, Tank, River Raid, Congo Bongo, Chiller, Booga-Boo. Todos en perfecto estado. Me gustaría tener el Night Shade, Profanation, Soccer. Envíar cinta a:

Javier Rosando López
Avda. de Concha Espina
Tel. (942) 70 05 16
39500 Cabezon de la Sal (Cantabria)

Compraría cartucho de expansión de memoria marca Sony HBM-64. Para contactar llamar:

Tel. (94) 681 49 30 (noches)
Durango (Vizcaya)

Intercambio programas y juegos MSX de todo tipo, poseo más de 100 (también cartuchos). Llamar a:

Juan Manuel
Tel. (954) 66 39 80
Sevilla

Vendo o Cambio juegos MSX, tengo más de cien títulos como King's Valley, Decathlon, Yier Ar Kunfu 1 y 2, Soccer, Star Avenger, Fórmula 1. Preguntar por:

Manolo
Tel. (954) 63 21 94
Sevilla

Intercambio programas de todo tipo MSX (excepto de revistas). Más de 150 títulos. Escribir a:

José Mª Munguía
Marqués de S. Esteban 42, 5ºA
Tel. (985) 34 04 77
33206 Gijón

Vendo cartuchos de juegos siguientes: Hiper Sport 2, Track & Field I, Super Tennis, Eddy II. Todos en 15.000 ptas.

Julio Leal Ruiz
Comandante Paz Varela s/.
Edificio Parque Feria, 7º C
Tel. 31 15 15
Jerez de la Frontera (Cádiz)



MSX: EL MANUAL ESCOLAR

Autor: Warner Voss
Editor: Ferre Moret
Páginas: 388
Precio: 2.800 ptas.

Intenta este libro de una forma didáctica ofrecer una panorámica de las posibilidades que nuestro ordenador **MSX** puede brindarnos dentro de diversos campos.

El libro comienza con una introducción para iniciar al lector en los comandos esenciales del lenguaje BASIC que necesitará según vaya avanzando en la lectura.

Se pasa a continuación a los capítulos de aplicaciones, en los que el autor nos da una visión del uso que podemos hacer en las áreas de matemáticas, química, física, idiomas, biología, ecología, geografía, historia y economía.

Cada capítulo está dividido en una consideración previa, a la que siguen una serie de programas concretos que sirven de ejemplo de la aplicación considerada.

En la consideración previa el autor trata de una manera general el sentido que tiene el uso del ordenador en el ámbito concreto del capítulo. Esta introducción es muy breve, pasándose inmediatamente a los programas.

La presentación de los programas es muy didáctica y está completamente estructurada en 7 pasos.

En el primero de ellos se da una breve presentación del problema que se pretende resolver. Seguidamente se hace un análisis del problema, llegando a los resultados que se pretenden obtener y a la forma de conseguirlos. En el tecer paso se da el diagrama de flujo u organigrama de la resolución del problema, para después dar un listado del mismo.

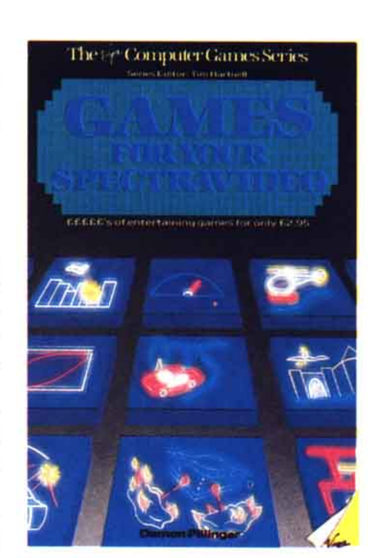
A continuación hay una lista de las variables utilizadas por el programa y del cometido de cada una, para comentar seguidamente una a una todas las líneas del listado, terminando con una explicación de cómo se presentan los resultados.

Como vemos, un ordenamiento tan completo de cada problema hace que este libro sea muy adecuado como instrumento de enseñanza y de autoaprendizaje del lenguaje BASIC y de las aplicaciones de nuestro ordenador **MSX**.

Por todo ello, **MSX, el manual escolar** es un libro altamente recomendable para el principiante en estos menesteres y, en cualquier caso, muy ameno y curioso para cualquier lector interesado en saber qué posibilidades le ofrece realmente su ordenador.

GAMES FOR YOUR SPECTRAVIDEO

Autor: Damon Pillinger y Danny Olesh
Editor: Rama
Páginas: 124
Precio: 1.000 ptas.



Encontramos en este libro otro de los muchos compendios de juegos y sencillas utilidades, encaminados a que nosotros mismos las introduzcamos en el ordenador y las grabemos, para incorporarlas posteriormente a nuestra biblioteca.

Este libro presenta una serie de 27 pequeños programas, principalmente de juegos, escritos en BASIC. La descripción de cada juego consiste en un pequeño comentario inicial, en el que se describe de forma muy sucinta el objetivo de cada programa, para pasar después a dar el listado del mismo.

A nuestro juicio, los programas contenidos en el libro no son de excesiva calidad, existiendo algunos (dibujo de círculos, dibujo de formas de onda), que después de ser ejecutados una vez y observado el efecto visual, se nos antojan de poca utilidad posterior.

Por otra parte, el pequeño comentario que se hace al comienzo del programa podría ser más amplio, ya que , aunque los programas son sencillos en su mayoría, el lector podría aprender mucho más si se le explicara con detalle cómo están hechos.

No obstante, entre los 27 programas que componen este libro existen algunos conseguidos y entretenidos. Podría haberse dedicado más espacio para mejorar éstos (incluyendo por ejemplo, rutinas en código máquina para mayor rapidez) en lugar de introducir muchos otros pequeños programas de peor calidad.

En resumen, **Games for your Spectravideo** puede ser calificado como un compendio más de juegos sencillos para nuestro ordenador. Dada la sencillez de los programas que contiene, va destinado principalmente a aquellas personas que se inician en el manejo de un ordenador **MSX**.

DESCUBRE TU MSX

Autor: Joe Pritchard
Editor: Anaya
Páginas: 232
Precio: 1.272

Descubre tu MSX es un libro bastante completo que explica de forma clara y concisa todo lo que el usuario de un ordenador del estándar saber, para sacar el máximo partido de su máquina.

El desarrollo de la obra es gradual, partiendo de una introducción al estándar **MSX** y de las sentencias esenciales del BASIC, y termina con unas nociones de código máquina, dando la información necesaria para que cualquier lector pueda hacer sus primeros «pinitos» con este tipo de lenguaje.

Entre un tema y otro se tocan todas las materias que tienen que ver con la programación de nuestro ordenador, tales como estructuras de datos, almacenamiento de datos y programas en *cassette*, formato de variables, gráficos, sonido, programación de juegos, etc.



Se da asimismo algunas rutinas sencillas que el lector puede usar en sus propios programas, ahorrándose tiempo y trabajo.

Un capítulo interesante es el dedicado al mapa de memoria del **MSX**. Este capítulo será, sin duda, bien recibido por la inestimable ayuda que nos ofrecerá en muchas de nuestras aplicaciones.

En definitiva, **Descubre tu MSX** es un estupendo libro de consulta, que no debería faltar en la biblioteca de ningún usuario de este tipo de ordenadores, ya que puede servir como libro de consulta al programador experimentado y como libro de estudio para todo el que quiera dar los primeros pasos en programación. Todo ello gracias al rigor, exactitud y sencillez a la hora de tratar los temas incluidos.

MSX

¡NO HAY COMPETENCIA POSIBLE!

en Calidad / Precio





DRO SOFT

MSX
AMSTRAD
SPECTRUM



MOLECULE MAN

Perdido en un laberinto de 256 habitaciones lucha en contra del tiempo y las radiaciones letales para teleportarte a lugar seguro. También incluido en esta cinta un único y fácil de utilizar, sistema de construcción de laberintos, que te permitirá corregir el existente o crear otros nuevos.

MSX
AMSTRAD
SPECTRUM

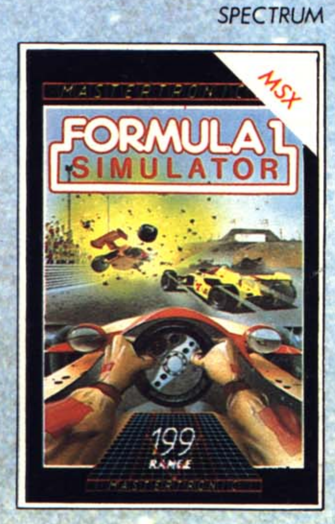


KNIGHT TYME

La tercera de la serie de aventuras Magic Knight, se encontró a sí mismo transportado al siglo 25 abordo de la nave estelar PISCIS. El juego utiliza un sistema mejorado de animación, que fue utilizado por primera vez en Spellbound. ¿Será éste el fin de Magic Knight?.

SERIE M.A.D.: P.V.P. 1.100 PTAS.

MSX
AMSTRAD
SPECTRUM



FORMULA 1

El juego de competición de mayor realismo, con los diez circuitos más famosos; SILVERSTONE, MONACO, MONZA... etc.

¡3, 2, 1... Adelante!

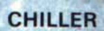
MSX
AMSTRAD
COMMODORE

CHILLER

MSX

199

RANGE

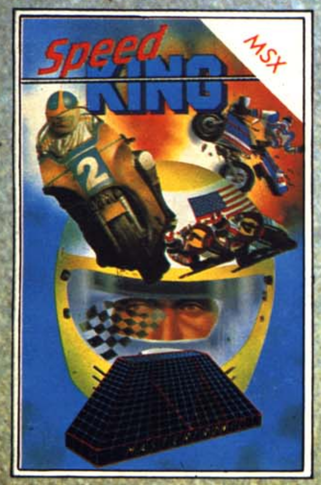


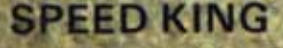
CHILLER

En una fría noche de Luna llena intentarás salvar a tu chica enfrentándote a cadáveres vivientes, arañas, espectros y murciélagos.

¡Animo y recoge todas las cruces que puedas!

MSX
COMMODORE





SPEED KING

El juego de carreras de Motocicletas con la emocionante acción de correr rueda con rueda contra otros 19 pilotos ¡Ponte el casco y vive la inolvidable aventura de las motos de altas prestaciones compitiendo a 250 millas a la hora!



SPACE WALK

Eres un astronauta al mando de la Lanzadora Espacial. Desde tu base en la luna vigilas los satelites descarriados y tienes que recuperarlos. Trabaja por la superficie de la luna y cuando sea necesario utiliza el JET-PACK para propulsarte al satelite.

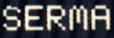


Licencia exclusiva para ESPAÑA DRO SOFT
Fundadores, 3 - 28028-MADRID
Tels. 255 45 00/09

YA A LA VENTA

tus primeros programas de apuestas

EN TARJETA







1X2 LOTO QH





1. TARJETA SOFTCARD 2.

TARJETA SOFTCARD

1. FORMATO PRESENTACION
2. ADAPTADOR PARA «MSX»

COMPRANDO LAS TARJETAS 1X2, QH O LOTO ADAPTADOR GRATIS

A LA VENTA EN TODOS LOS ESTABLECIMIENTOS DE

Y EN TODOS LOS DISTRIBUIDORES DE NUESTROS PRODUCTOS

